TABLE OF CONTENTS

CHAPTER 1:

INTRODUCTION

1.INTRODUCTION

1.1 Problem Definition

Security service companies face numerous hurdles in effectively managing their workforce and streamlining payroll procedures. These challenges stem from manual record-keeping practices, intricate deployment processes, and the complexities involved in payroll calculations. Such inefficiencies not only lead to errors but also significantly escalate administrative burdens. The need of the hour is a comprehensive solution that can alleviate these pain points and enhance operational efficiency.

1.2 Problem Description

This project aims to develop a Security Services Management Software tailored for security service companies. The software will address the challenges these companies face in managing their workforce and automating payroll processes. By eliminating manual record-keeping, simplifying deployment procedures, and automating payroll calculations, the software will enhance operational efficiency and reduce administrative burdens. It will provide a centralized platform for organizing workforce schedules, tracking employee activities, and generating accurate payroll reports. This solution seeks to streamline operations and empower security service companies to focus more on delivering high-quality security services to their clients.

CHAPTER 2:

SYSTEM STUDY

1. SYSTEM STUDY

2.1 Existing System

The security service industry currently faces challenges in managing workforce deployment and automating payroll processes efficiently. The reliance on manual methods for record-keeping and payroll calculations introduces potential for inefficiency and error, imposing significant administrative burdens. Existing systems lack the integration of comprehensive workforce management with automated payroll functionalities, hindering operational efficiency.

2.2 Feasibility Study

The basic idea behind feasibility study is to determine whether the project is feasible or not. Feasibility is conducted to identify a best system that meets all the requirements. This includes an identification, description, an evaluation of the proposed systems and selection of the best system for the job. The requirements of the system are specified with a set of constraints such as system objectives and the description of the out puts. It is then duty of the analyst to evaluate the feasibility of the proposed system to generate the above results. Three key factors are to be considered during the feasibility study.

2.3.1 Operational Feasibility

The software is developed with a focus on user-friendliness, utilizing VB.NET for the frontend to ensure easy navigation and data management. This section evaluates how the software will integrate into daily operations, the extent of training required for users, and the organization's readiness for adoption. The aim is to minimize operational disruptions and enhance efficiency through the software's implementation.

2.3.2 Technical Feasibility

The technical feasibility of the Security Services Management Software was rigorously evaluated, focusing on the organization's technological capabilities to support the system. This evaluation particularly considered performance, reliability, maintainability, and productivity. Choosing SQL Server Management Studio for the backend is pivotal for its superior data management features, ensuring the system's robustness and efficiency.

A pre-development review of the organization's technological resources revealed adequate computing facilities and sophisticated hardware, affirming the project's technical viability with SQL Server Management Studio.

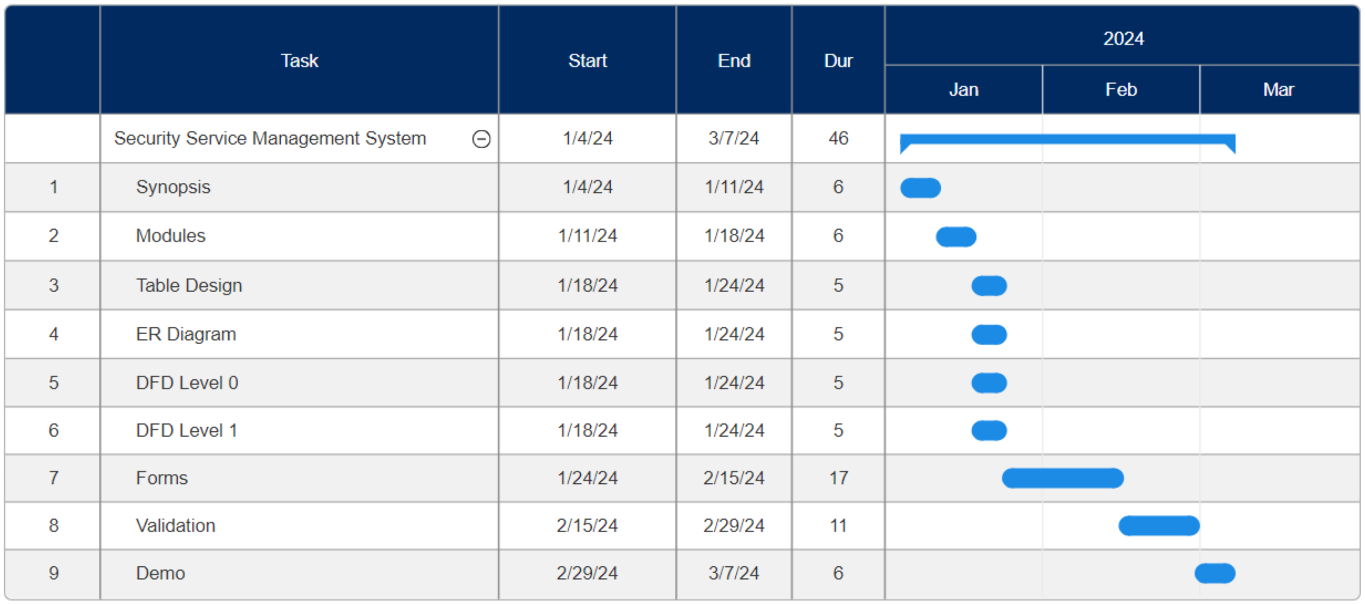
2.3.3 Economic Feasibility

Economic feasibility is critical in assessing the Security Services Management Software's viability, emphasizing the analysis's pivotal role in determining the project's financial soundness. This stage involves a detailed cost-benefit analysis, aimed at evaluating the economic implications of introducing the software within the organization's operations. The primary focus is on achieving a favorable balance between the investment needed for software development and deployment, and the anticipated operational efficiencies, cost savings, and enhanced client satisfaction. Given the organization's readiness in terms of necessary infrastructure, the project is anticipated to offer substantial economic benefits, affirming its financial feasibility.

2.3 Proposed System

To overcome these challenges, the Security Services Management Software offers a sophisticated solution designed to streamline the deployment of security personnel and automate payroll processes. This system enables detailed management of worker profiles, client information, deployment specifics, and payroll details. A key feature of this software is its automated payroll generation, which calculates compensation based on unique worker codes, the number of days worked, and the specific site of deployment, ensuring timely and accurate payroll processing without the need for manual intervention.

2.4 Gantt Chart



	Task	Start	End	Dur
	Security Service Management System	1/4/24	3/7/24	46
1	Synopsis	1/4/24	1/11/24	6
2	Modules	1/11/24	1/18/24	6
3	Table Design	1/18/24	1/24/24	5
4	ER Diagram	1/18/24	1/24/24	5
5	DFD Level 0	1/18/24	1/24/24	5
6	DFD Level 1	1/18/24	1/24/24	5
7	Forms	1/24/24	2/15/24	17
8	Validation	2/15/24	2/29/24	11
9	Demo	2/29/24	3/7/24	6

CHAPTER 3:

SYSTEM DESIGN

3. ER DIAGRAM, DFD [LVL 0, LVL 1]

3.1 ER Diagram:

Entity relationship model defines the conceptual view of database. It works around real-world entity and association among them. At view level, ER model is considered well for designing databases.

Entity:

A real-world thing either animate or inanimate that can be easily identifiable and distinguishable. For example, in a school database, student, teachers, class and course offered can be considered as entities. All entities have some attributes or properties that give them their identity

An entity set is a collection of similar types of entities. Entity set may contain entities with attribute sharing similar values. For example, Students set may contain all the student of a school; likewise, Teachers set may contain all the teachers of school from all faculties. Entities sets need not to be disjoint

Attributes:

Entities are represented by means of their properties, called attributes. All attributes have values. For example, a student entity may have name, class, age as attributes.

There exists a domain or range of values that can be assigned to attributes. For example, a student's name cannot be a numeric value. It has to be alphabetic. A student's age cannot be negative, etc.

An entity–relationship model (ER model) describes inter-related things of interest in a specific domain of knowledge. An ER model is composed of entity types (which classify the things of interest) and specifies relationships that can exist between instances of those entity types.

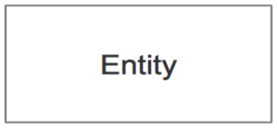
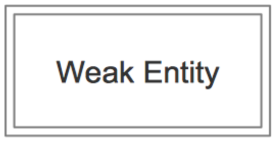
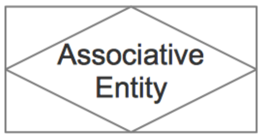
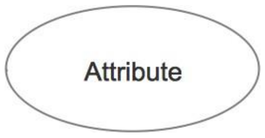
In software engineering an ER model is commonly formed to represent things that a business needs to remember in order to perform business processes. Consequently, the ER model becomes an abstract data model that defines a data or information structure that can be implemented in a database, typically a relational database

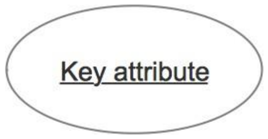
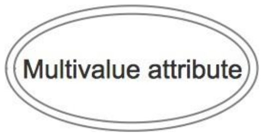
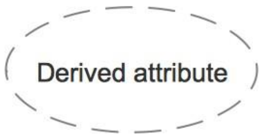
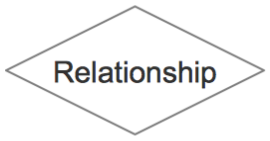
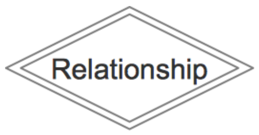
Entity–relationship modeling was developed for database design by Peter Chen and published in a 1976 paper. However, variants of the idea existed previously, some ER modelers show super and subtype entities connected by generalization-specialization relationships, and an ER model can be used also in the specification of domain-specific ontology.

<u>3.1.1 Steps – How to Draw ER Diagram –</u>

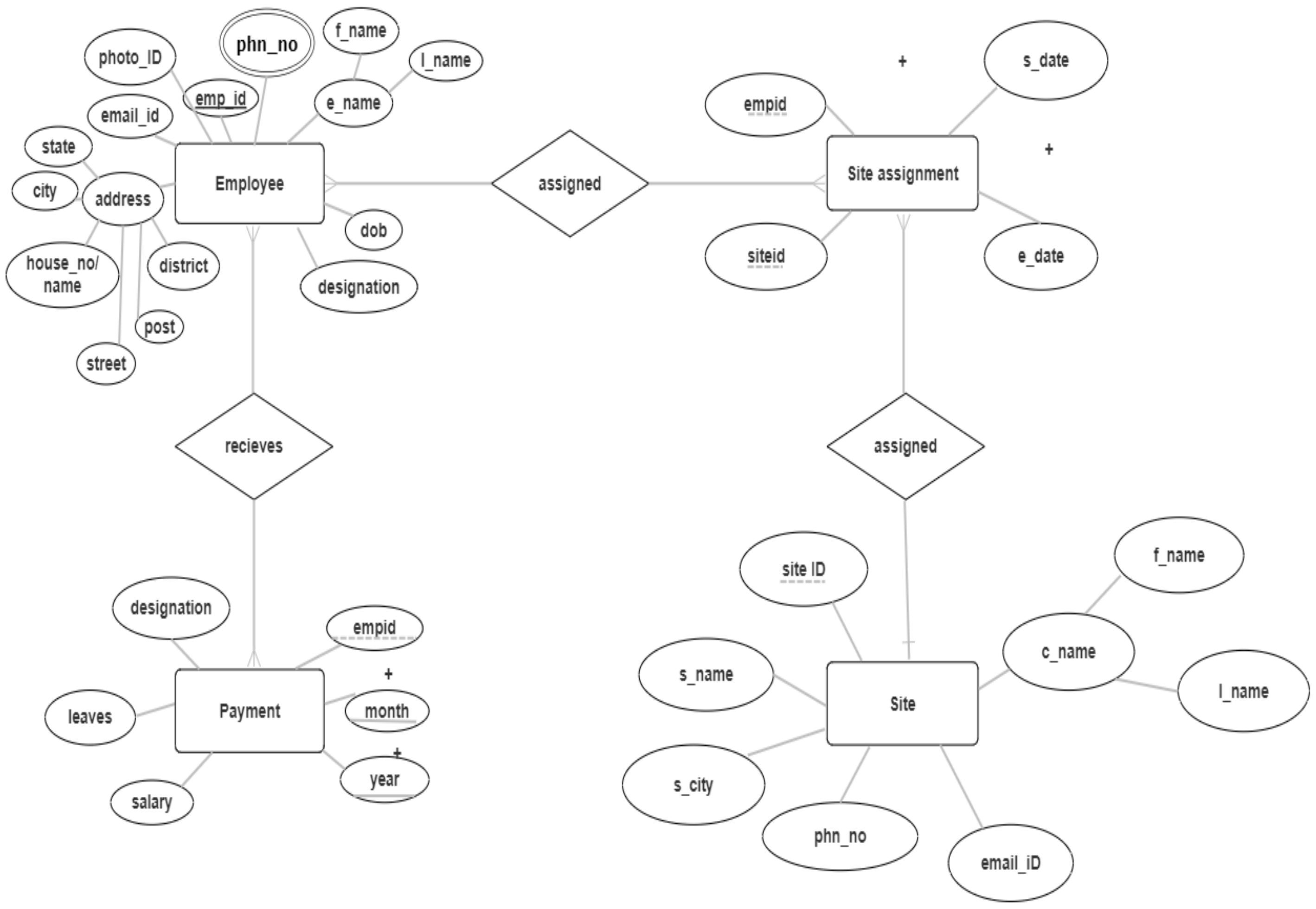
1. Identify all the entities of the given problem.
2. Identify all the attributes of the entities identified in step 1.
3. Identify the Primary Keys of entities identified in Step 1.
4. Identify the Attribute Types of attributes identified in step 2
5. Identify relationship between the entities and constraints on the entities and implement them.

3.1.2 Entity Relationship Diagram Symbols and descriptions:

Symbol	**Name**	**Description**
	Entity	An entity is represented by a rectangle which contains the entity's name.
	Weak entity	An entity that cannot be uniquely identified by its attributes alone. The existence of a weak entity is dependent upon another entity called the owner entity. The weak entity's identifier is a combination of the identifier of the owner entity and the partial key of the weak entity.
	Associative Entity	An entity used in a many-to- many relationship (represents an extra table). All relationships for the associative entity should be many
	Attribute	In the Chen notation, each attribute is represented by an oval containing attribute's name

	Key Attribute	An attribute that uniquely identifies a particular entity. The name of a key attribute is underscored.
	Multi-value Attribute	An attribute that can have many values (there are many distinct values entered for it in the same column of the table). Multi- valued attribute is depicted by a dual oval.
	Derived Attribute	An attribute whose value is calculated (derived) from other attributes. The derived attribute may or may not be physically stored in the database. In the Chen notation, this attribute is represented by dashed oval.
	Strong Relationship	A relationship where entity is existence-independent of other entities and PK of Child doesn't contain PK component of Parent Entity. A strong relationship is represented by a single rhombus
	Weak Relationship	A relationship where Child entity is existence-dependent on parent and PK of Child Entity contains PK component of Parent Entity. This relationship is represented by a double rhombus.

3.1.3 ER Diagram of Security Service Management System:



3.2 Data Flow Diagram:

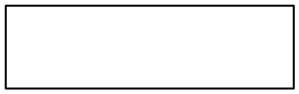
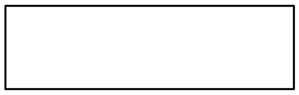
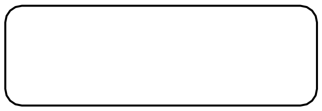
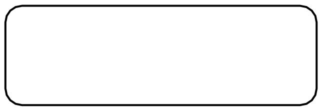
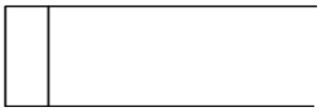
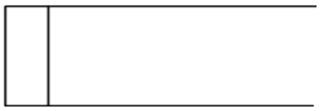
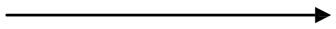
A data flow diagram (DFD) is a graphical representation of the "flow" of data through an information system, modelling its process aspects. A DFD is often used as a preliminary step to create an overview of the system, which can later be elaborated. DFDs can also be used for the visualization of data processing (structured design).

A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel (which is shown on a flowchart).

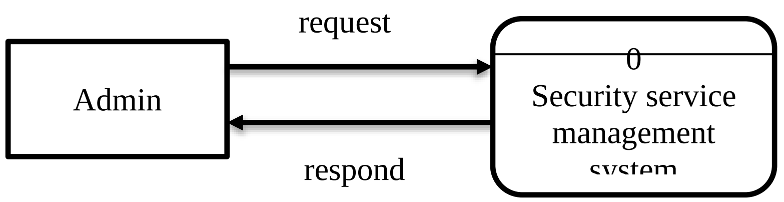
A DFD shows what kind of information will be input to and output from the system, where the data will come from and go to, and where the data will be stored. It does not show information about the timing of process or information about whether processes will operate in sequence or in parallel

It is common practice to draw the context-level data flow diagram first, which shows the interaction between the system and external agents which act as data sources and data sinks. This helps to create an accurate drawing in the context diagram. The system's interactions with the outside world are modelled purely in terms of data flows across the system boundary. The context diagram shows the entire system as a single process, and gives no clues as to its internal organization

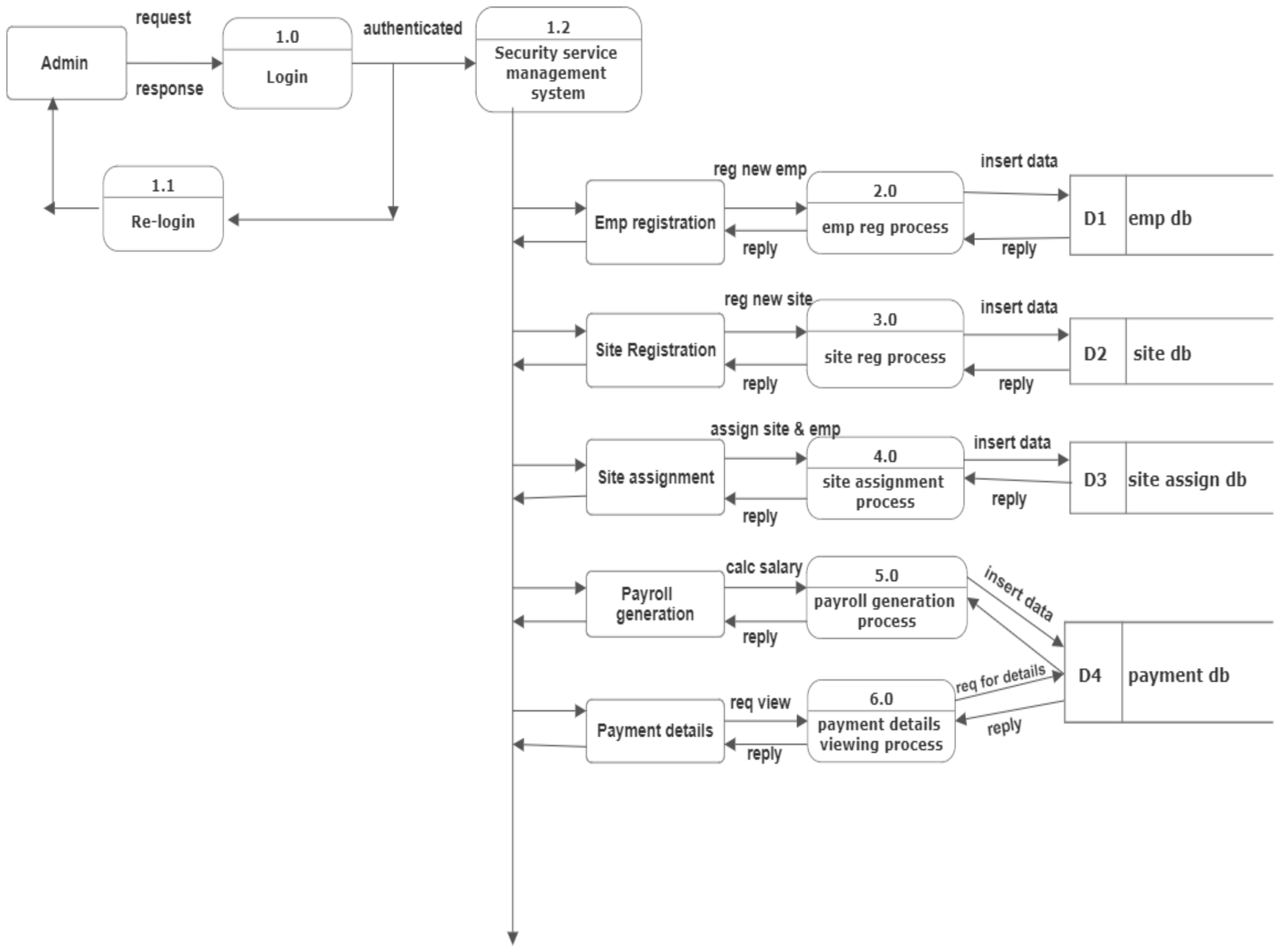
3.2.1 DFD Symbols and meanings:

Symbol	**Meaning**	**Example**
	An entity. A source of data or a destination for data.	 Employee
	A process or task that is performed by the system.	 Registration
	A data store, a place where data is held between processes.	 Employee Database
	A data flow	Authentication

3.2.2 DFD level 0:



3.2.2 DFD level 1:



CHAPTER 4:

SYSTEM CONFIGURATION

2. SYSTEM CONFIGURATION

4.1 HARDWARE REQUIREMENTS

Processor	intel i3(dual core) or above
Clock speed	3.00 GHz
Ram	4gb or more
Hdd	50gb
Monitor	SVGA color
Key board	101 keys
Mouse	ps2/ serial

4.2 SOFTWARE REQUIREMENTS

Operating System	Windows 10
Programming Language	VB.net
Database	SQL Server 2019
Tool	Visual Studio 2022

CHAPTER 5:

DETAILS OF SOFTWARE

5. DETAILS OF SOFTWARE

5.1 Overview of frontend

Introduction To the .Net Framework:

The .NET Framework is a new computing platform that simplifies application development in the highly distributed environment of the Internet. The .NET Framework is designed to fulfil the following objectives:

- To provide a consistent object-oriented programming environment whether object code is stored and executed locally, executed locally but Internet-distributed, or executed remotely.
- To provide a code-execution environment that minimizes software deployment and versioning conflicts.
- To provide a code-execution environment that guarantees safe execution of code, including code created by an unknown or semi-trusted third party.
- To provide a code-execution environment that eliminates the performance problems of scripted or interpreted environments.
- To make the developer experience consistent across widely varying types of applications, such as Windows-based applications and Web-based applications.
- To build all communication on industry standards to ensure that code based on the

  .NET Framework can integrate with any other code.

The .NET Framework has two main components: the common language runtime and the .NET Framework class library. The common language runtime is the foundation of the .NET Framework. You can think of the runtime as an agent that manages code at execution time, providing core services such as memory management, thread management while also enforcing strict type safety and other forms of code accuracy that ensure security and robustness. In fact, the concept of code management is a fundamental principle of the runtime. Code that targets the runtime is known as managed code, while code that does not target the runtime is known as unmanaged code. The class library, the other main component of the .NET Framework, is a comprehensive, object-oriented collection of reusable types that you can use to develop applications ranging from traditional command-line or graphical user interface (GUI) applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services

5.2 Overview of backend

5.2.1 SQL server:

A database management, or DBMS, gives the user access to their data and helps them transform the data into information. Such database management systems include dBase, paradox, IMS, SQL Server and SQL Server. These systems allow users to create, update and extract information from their database.

A database is a structured collection of data. Data refers to the characteristics of people, things and events. SQL Server stores each data item in its own fields. In SQL Server, the fields relating to a particular person, thing or event are bundled together to form a single complete unit of data, called a record (it can also be referred to as raw or an occurrence). Each record is made up of a number of fields. No two fields in a record can have the same field name.

During an SQL Server Database design project, the analysis of your business needs identifies all the fields or attributes of interest. If your business needs change over time, you define any additional fields or change the definition of existing fields

5.2.2 SQL server tables:

SQL Server stores records relating to each other in a table. Different tables are created for the various groups of information. Related tables are grouped together to form a database.

5.2.3 Primary key:

Every table in SQL Server has a field or a combination of fields that uniquely identifies each record in the table. The Unique identifier is called the Primary Key, or simply the Key. The primary key provides the means to distinguish one record from all other in a table. It allows the user and the database system to identify, locate and refer to one particular record in the database.

5.2.4 Relational database:

Sometimes all the information of interest to a business operation can be stored in one table. SQL Server makes it very easy to link the data in multiple tables. Matching an employee to the department in which they work is one example. This is what makes SQL Server a relational database management system, or RDBMS. It stores data in two or more tables and enables you to define relationships between the tables and enables you to define relationships between the tables.

5.2.5 Foreign key:

When a field is one table matches the primary key of another field is referred to as a foreign key. A foreign key is a field or a group of fields in one table whose values match those of the primary key of another table

5.2.6 Referential integrity:

Not only does SQL Server allow you to link multiple tables, it also maintains consistency between them. Ensuring that the data among related tables is correctly matched is referred to as maintaining referential integrity

5.3 System Maintenance and Evaluation:

System maintenance is the process of modifying software product after it is delivered to the customer. Software maintenance is inevitable as it is subjected to wear and tear.

- It is necessary to rectify some errors in the system or increase the performance of the system. This is done by corrective maintenance.
- If the customer demand that their product has to run on new platform then adoptive maintenance is required. It is also applied when the system has to interface with new hardware or software.

- It helps to support new system features. The users can change different functions of the system.

The project is developed using the Water fall model. It suggests the systematic and sequential approach to software development. Waterfall model is the oldest and most widely used process model. Each phase of the life cycle is completed before the start of a new phase. It is the engineering approach. The waterfall model makes the project:

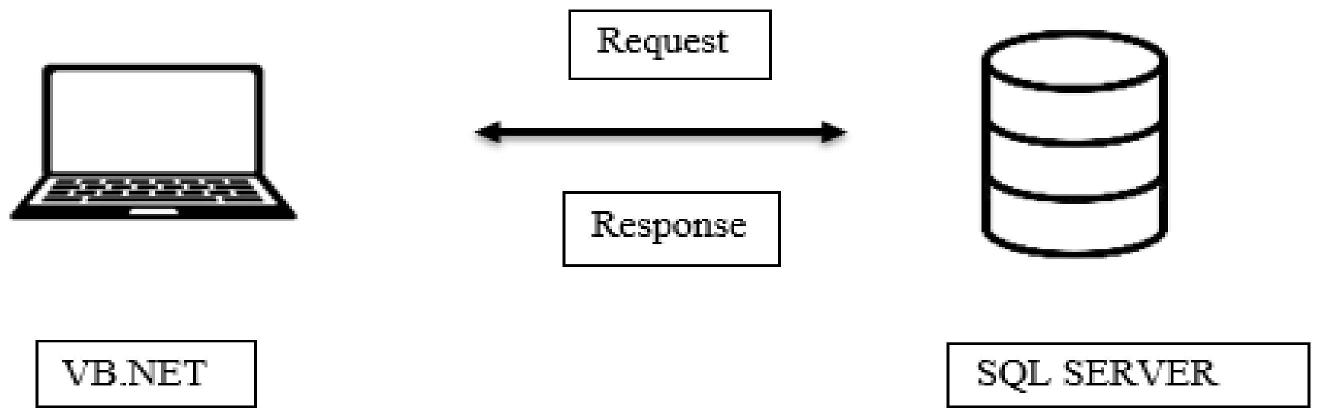
- Simple in nature.
- Easy to implement.
- Systematic approach.
- It is an essential approach to software development

CHAPTER 6:

SYSTEM DESIGN

6. SYSTEM DESIGN

6.1 Architectural Design



6.2 Input /Output Design:

The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy.

6.2.1 Input Design:

Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in a maize of instant. Thus the objective of input design is to create an input layout that is easy to follow.

Below are VB.NET codes used in the development of the project:

Employee Registration Page

```
Imports System.Globalization
Imports System.Text.RegularExpressions
Imports System.Windows.Forms.VisualStyles.VisualStyleElement
Imports System.Data
Imports System.Data.SqlClient

Public Class Form4

    Dim connectionString As String = "Data Source=DESKTOP-
O16HO1G\SQLEXPRESS;Initial Catalog=Security Service Management
System;Integrated Security=True;"

    'Function to generate new Empid
    Private Function GenerateNextEmpIDFromDatabase() As String
        Dim nextEmpID As String = "SG01" ' Default value

        Try
            ' SQL query to get the last employee ID from the database
            Dim queryLastEmpID As String = "SELECT TOP 1 empid FROM Emp_table
ORDER BY CAST(SUBSTRING(empid, 3, LEN(empid)) AS INT) DESC"

            Using connection As New SqlConnection(connectionString)
                Using command As New SqlCommand(queryLastEmpID, connection)

                    connection.Open()
                    Dim lastEmpID As String =
Convert.ToString(command.ExecuteScalar())

                    ' If there are records in the database, generate the next employee ID
                    If Not String.IsNullOrEmpty(lastEmpID) Then
                        ' Extract the numeric part of the last employee ID
                        Dim numericPart As String = lastEmpID.Substring(2)
                        ' Convert the numeric part to an integer and increment by 1

                        Dim nextNumericPart As Integer = Convert.ToInt32(numericPart) + 1
                        ' Generate the next employee ID by combining the prefix ("SG") and
the incremented numeric part
                        nextEmpID = "SG" & nextNumericPart.ToString("D2")
                    End If
                End Using
            End Using
        Catch ex As Exception
            MessageBox.Show("Error generating next employee ID: " & ex.Message,
"Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
        End Try
        Return nextEmpID
    End Function

    Private Function IsValidEmail(email As String) As Boolean
        ' Regular expression pattern for email validation
        Dim pattern As String = "^[a-z0-9](?:[a-z0-9](?!\.)|\.(?![^.])){3,18}[a-z0-9]@[a-
    z0-9-]+\.(?:com|net|org)$"
        ' Check if the email matches the pattern
        Return Regex.IsMatch(email, pattern)
    End Function

    'Function to validate whether the phone number is a repeating number upto 6 digits
    Private Function ContainsRepeatingSequence(phoneNumber As String) As
Boolean
        ' Remove the country code
        Dim numberWithoutCode As String = phoneNumber.Substring(3)
        ' Check if the remaining digits contain repeating sequences
        For i As Integer = 0 To numberWithoutCode.Length - 1
            Dim currentDigit As Char = numberWithoutCode(i)
```

```
            Dim sequenceLength As Integer = 1



            ' Check subsequent digits for repetition
            For j As Integer = i + 1 To numberWithoutCode.Length - 1
                If numberWithoutCode(j) = currentDigit Then

                    sequenceLength += 1
                    If sequenceLength = 6 Then ' A sequence of 6 repeating digits found
                        Return True
                    End If
                Else
                    Exit For
                End If
            Next j
        Next i

        ' No repeating sequences found
        Return False
    End Function

    ' Check if any character, number, or special character is repeated more than once
    Private Function ContainsRepeatedCharacters(input As String) As Boolean
        Dim count As Integer = 1
        For i As Integer = 1 To input.Length - 1
            If input(i) = input(i - 1) Then
                count += 1
                If count > 2 Then
                    Return True
                End If
            Else
                count = 1
            End If
        Next
        Return False
    End Function

    Private Sub Form4_Load(sender As Object, e As EventArgs) Handles
MyBase.Load

        ' Set the form's AcceptButton property to Button1
        Me.AcceptButton = Button1



        ' Emp ID to start with "SG"
        TextBox1.Text = "SG"

        ' Set initial text of phone number textbox to "+91"
        TextBox5.Text = "+91"

        ' Set default date for dob
        DateTimePicker1.Value = DateTime.Today

        ' Add photo ID types to the ComboBox
        ComboBox1.Items.Add("Aadhar Card")
        ComboBox1.Items.Add("Driver's License")
        ComboBox1.Items.Add("Passport")
        ComboBox1.Items.Add("Voter ID")

        ' Set a default selection if needed
        ComboBox1.SelectedIndex = 0

        ' Add states to the ComboBox
        ComboBox2.Items.Add("Andhra Pradesh")
        ComboBox2.Items.Add("Arunachal Pradesh")
        ComboBox2.Items.Add("Assam")
        ComboBox2.Items.Add("Bihar")
        ComboBox2.Items.Add("Chhattisgarh")
        ComboBox2.Items.Add("Goa")
        ComboBox2.Items.Add("Gujarat")
        ComboBox2.Items.Add("Haryana")
        ComboBox2.Items.Add("Himachal Pradesh")
        ComboBox2.Items.Add("Jharkhand")
        ComboBox2.Items.Add("Karnataka")
        ComboBox2.Items.Add("Kerala")
        ComboBox2.Items.Add("Madhya Pradesh")
        ComboBox2.Items.Add("Maharashtra")
        ComboBox2.Items.Add("Manipur")
        ComboBox2.Items.Add("Meghalaya")
        ComboBox2.Items.Add("Mizoram")

        ComboBox2.Items.Add("Nagaland")
        ComboBox2.Items.Add("Odisha")
        ComboBox2.Items.Add("Punjab")
        ComboBox2.Items.Add("Rajasthan")
        ComboBox2.Items.Add("Sikkim")
        ComboBox2.Items.Add("Tamil Nadu")
        ComboBox2.Items.Add("Telangana")
        ComboBox2.Items.Add("Tripura")
        ComboBox2.Items.Add("Uttar Pradesh")
        ComboBox2.Items.Add("Uttarakhand")
        ComboBox2.Items.Add("West Bengal")
        ComboBox2.Items.Add("Andaman and Nicobar Islands")
        ComboBox2.Items.Add("Chandigarh")
        ComboBox2.Items.Add("Dadra and Nagar Haveli")
        ComboBox2.Items.Add("Daman and Diu")
        ComboBox2.Items.Add("Lakshadweep")
        ComboBox2.Items.Add("Delhi")
        ComboBox2.Items.Add("Puducherry")
        ' Add different ranks to Designation
        ComboBox3.Items.Add("Security Guard")
        ComboBox3.Items.Add("Lady Guard")
        ComboBox3.Items.Add("Field Officer")
        ComboBox3.Items.Add("Supervisor")

        ' Reset designation ComboBox to its first default option
        ComboBox3.SelectedIndex = 0

        'Disable manual entry into comboBox
        ComboBox1.DropDownStyle = ComboBoxStyle.DropDownList
        ComboBox2.DropDownStyle = ComboBoxStyle.DropDownList
        ComboBox3.DropDownStyle = ComboBoxStyle.DropDownList

    End Sub



    Private Sub TextBox11_TextChanged(sender As Object, e As EventArgs) Handles
TextBox11.TextChanged
        If TextBox11.Text.Length > 0 Then
            ' Get the current selection start position
            Dim selectionStart As Integer = TextBox11.SelectionStart
            ' Capitalize the first letter
            TextBox11.Text = TextBox11.Text.Substring(0, 1).ToUpper() +
TextBox11.Text.Substring(1)
            ' Restore the selection start position
            TextBox11.SelectionStart = selectionStart
        End If
    End Sub

    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles
Button2.Click
        ' Close the current form (Form4) and open the first form (Form3) again
        Dim form3 As New Form3()
        form3.Show()
        Me.Close()

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

        ' Perform frontend validations
        If Not ValidateInputs() Then
            Return
        End If

        ' Check if empid already exists
        Dim empIdExists As Boolean = False
        Dim queryEmpIdExists As String = "SELECT COUNT(*) FROM Emp_table
WHERE empid = @empid"
        Try
            Using connection As New SqlConnection(connectionString)
                Using commandCheckEmpId As New SqlCommand(queryEmpIdExists,
connection)

                    connection.Open()
                    commandCheckEmpId.Parameters.AddWithValue("@empid",
TextBox1.Text)
 Dim countAsInteger = Convert.ToInt32(commandCheckEmpId.ExecuteScalar())
                    empIdExists = (count > 0)
                End Using
            End Using
        Catch ex As Exception
            MessageBox.Show("Error checking employee ID: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return
        End Try
        If empIdExists Then

            MessageBox.Show("Employee ID is already taken.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return

        End If
        ' Insert data into the database
        Dim query As String = "INSERT INTO [dbo].[Emp_table] ([empid], [efname],
[elname], [dob], [designation], [phn_no], [alt_phnno], [email], [photo_id],
[house_noname], [street_name], [city_name], [district_name], [state], [pincode]) " &
                              "VALUES (@empid, @efname, @elname, @dob, @designation,
@phn_no, @alt_phnno, @email, @photo_id, @house_noname, @street_name,
@city_name, @district_name, @state, @pincode)"
        Try
            Using connection As New SqlConnection(connectionString)
                Using command As New SqlCommand(query, connection)
                    ' Add parameters
                    command.Parameters.AddWithValue("@empid", TextBox1.Text)
                    command.Parameters.AddWithValue("@efname", TextBox2.Text)
                    command.Parameters.AddWithValue("@elname", TextBox3.Text)
                    command.Parameters.AddWithValue("@dob", DateTimePicker1.Value)
                    command.Parameters.AddWithValue("@designation",
ComboBox3.SelectedItem.ToString())

                    command.Parameters.AddWithValue("@phn_no", TextBox5.Text)
                    command.Parameters.AddWithValue("@alt_phnno",
If(String.IsNullOrEmpty(TextBox6.Text), DBNull.Value, TextBox6.Text))
                    command.Parameters.AddWithValue("@email",
If(String.IsNullOrEmpty(TextBox7.Text), DBNull.Value, TextBox7.Text))
                    command.Parameters.AddWithValue("@photo_id",
ComboBox1.SelectedItem.ToString())
                    command.Parameters.AddWithValue("@house_noname",
TextBox9.Text)
```

```
                command.Parameters.AddWithValue("@street_name", TextBox10.Text)

                command.Parameters.AddWithValue("@city_name", TextBox11.Text)
                command.Parameters.AddWithValue("@district_name",
TextBox12.Text)
                command.Parameters.AddWithValue("@state",
ComboBox2.SelectedItem.ToString())
                command.Parameters.AddWithValue("@pincode", TextBox14.Text)

                connection.Open()

                command.ExecuteNonQuery()
                MessageBox.Show("Employee Registered Successfully",
"Confirmation", MessageBoxButtons.OK)
                ClearInputs()
            End Using
        End Using
    Catch ex As Exception
        MessageBox.Show("Error inserting data: " & ex.Message, "Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
    End Try
  End Sub

  Private Function ValidateInputs() As Boolean ' Frontend validations

    Dim regexEmpID As New Regex("^SG\d+$")
    ' Employee ID Presence Check
    If String.IsNullOrWhiteSpace(TextBox1.Text) Then
        MessageBox.Show("Employee ID cannot be blank", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
        TextBox1.Focus()
        Return False
    ElseIf Not regexEmpID.IsMatch(TextBox1.Text) Then
        MessageBox.Show("Invalid EMPLOYEE ID", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
```

```
            TextBox1.Focus()
            Return False

        End If
        'First and Last Name Presence check
        If String.IsNullOrWhiteSpace(TextBox2.Text) OrElse
String.IsNullOrWhiteSpace(TextBox3.Text) Then
            MessageBox.Show("Both first name and last name are required.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Return False

        ElseIf Not TextBox2.Text.All(Function(c) Char.IsLetter(c) Or c = " ") Then
            MessageBox.Show("Invalid first name", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox2.Focus()
            Return False
        ElseIf ContainsRepeatedCharacters(TextBox2.Text) Then

            MessageBox.Show("First name should not contain repeated characters",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox2.Focus()
            Return False
        End If

        'Last Name Presence check
        If Not TextBox3.Text.All(Function(c) Char.IsLetter(c) Or c = " ") Then
            MessageBox.Show("Invalid last name", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox3.Focus()
            Return False
        ElseIf ContainsRepeatedCharacters(TextBox3.Text) Then

            MessageBox.Show("Last name should not contain repeated characters",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox3.Focus()
            Return False
        End If
        ' Date Presence check
        If DateTimePicker1.Value = DateTime.Today Then
            MessageBox.Show("Please enter a valid DOB", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            DateTimePicker1.Focus()
            Return False
        ElseIf (DateTimePicker1.Value > DateTime.Today) Then
            MessageBox.Show("DOB cannot be in future", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            DateTimePicker1.Focus()
            Return False
        ElseIf (DateTime.Today - DateTimePicker1.Value).TotalDays < (18 * 365.25)
Then
            MessageBox.Show("Employee must be atleast 18 years old to register",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            DateTimePicker1.Focus()
            Return False
        ElseIf (DateTime.Today - DateTimePicker1.Value).TotalDays > (60 * 365.25)
Then
            MessageBox.Show("Employee cannot be above 60 years old to register",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            DateTimePicker1.Focus()
            Return False
        End If


        ' Phone Number Presence check
        If String.IsNullOrWhiteSpace(TextBox5.Text) Then
            MessageBox.Show("Phone number cannot be blank", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox5.Focus()
            Return False
        ElseIf TextBox5.Text = "+91" Then
```

```
            MessageBox.Show("Please enter phone number ", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)


            TextBox5.Focus()
            Return False
        ElseIf Not TextBox5.Text.StartsWith("+91") OrElse TextBox5.Text.Length <>
13 OrElse Not TextBox5.Text.Substring(3).All(Function(c) Char.IsDigit(c)) OrElse
TextBox5.Text.Substring(3, 1) Like "[0-6]" Then
            MessageBox.Show("Invalid Phone number", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox5.Focus()
            Return False
        ElseIf ContainsRepeatingSequence(TextBox3.Text) Then
            MessageBox.Show("Phone number contains a repeating sequence of more
than 6 digits", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox3.Focus()
            Return False
        End If


        ' Alternate Phone Number Presence and Format check
        If Not String.IsNullOrEmpty(TextBox6.Text) Then
            If TextBox6.Text = "+91" Then
                MessageBox.Show("Please enter alternate number ", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                TextBox6.Focus()
                Return False
            ElseIf Not TextBox6.Text.StartsWith("+91") OrElse TextBox6.Text.Length
<> 13 OrElse Not TextBox6.Text.Substring(3).All(Function(c) Char.IsDigit(c))
OrElse TextBox6.Text.Substring(3, 1) Like "[0-6]" Then
                MessageBox.Show("Invalid alternate phone number", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                TextBox6.Focus()
                Return False
            ElseIf TextBox5.Text = TextBox6.Text Then
```

```
                MessageBox.Show("Phone number and alternate phone number cannot be
the same", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                TextBox5.Focus()
                Return False
            ElseIf ContainsRepeatingSequence(TextBox6.Text) Then
                MessageBox.Show("Invalid Alternate Phone Number", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
                TextBox6.Focus()
                Return False
            End If
        End If
        ' Email format check
        If Not String.IsNullOrWhiteSpace(TextBox7.Text) AndAlso Not
IsValidEmail(TextBox7.Text) Then
            MessageBox.Show("Invalid email ID. Please enter a valid email address.",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox7.Focus()
            Return False
        End If
        ' House no/name Presence and Format check
        If String.IsNullOrWhiteSpace(TextBox9.Text) Then
            MessageBox.Show("Please enter house/building name or number",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox9.Focus()
            Return False
        ElseIf ContainsRepeatedCharacters(TextBox9.Text) Then
            MessageBox.Show("House/building name or number should not contain
repeated characters, numbers, or special characters", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox9.Focus()
            Return False
        End If
        ' Street Presence and Format check
        If String.IsNullOrWhiteSpace(TextBox10.Text) Then
```

```
            MessageBox.Show("Please enter street", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox10.Focus()
            Return False
        ElseIf ContainsRepeatedCharacters(TextBox10.Text) Then


            MessageBox.Show("Street should not contain repeated characters, numbers,
or special characters", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)


            TextBox10.Focus()
            Return False
        End If


        ' City Presence and Format check
        If String.IsNullOrWhiteSpace(TextBox11.Text) Then
            MessageBox.Show("Please enter the City", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox11.Focus()
            Return False
        ElseIf Not TextBox11.Text.All(Function(c) Char.IsLetter(c) Or c = " ") Then
            MessageBox.Show("Invalid City name", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox11.Focus()
            Return False
        ElseIf ContainsRepeatedCharacters(TextBox11.Text) Then
            MessageBox.Show("City should not contain repeated characters, numbers, or
special characters", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            TextBox11.Focus()
            Return False
        End If


        ' District Presence and Format check
        If String.IsNullOrWhiteSpace(TextBox12.Text) Then
```

```
            MessageBox.Show("Please enter the District", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox12.Focus()


            Return False
        ElseIf Not TextBox12.Text.All(Function(c) Char.IsLetter(c) Or c = " ") Then
            MessageBox.Show("Invalid District name", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)


            TextBox12.Focus()
            Return False
        ElseIf ContainsRepeatedCharacters(TextBox12.Text) Then
            MessageBox.Show("District should not contain repeated characters, numbers,
or special characters", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            TextBox12.Focus()
            Return False
        End If
        ' Check if State is selected from ComboBox2
        If ComboBox2.SelectedIndex = -1 Then
            MessageBox.Show("Please select the State", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            ComboBox2.Focus()
            Return False
        End If


        ' Pincode Presence and Format check
        If String.IsNullOrWhiteSpace(TextBox14.Text) Then
            MessageBox.Show("Please enter the Pincode", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox14.Focus()
            Return False
        ElseIf TextBox14.Text.Length <> 6 OrElse Not TextBox14.Text.All(Function(c)
Char.IsDigit(c)) Then
            MessageBox.Show("Invalid Pincode", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox14.Focus()
            Return False
        ElseIf TextBox14.Text.StartsWith("0") Then
            MessageBox.Show("Pincode should not start with '0'", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)

            TextBox14.Focus()
            Return False
        ElseIf TextBox14.Text.GroupBy(Function(c) c).Any(Function(g) g.Count() > 3)
Then

            MessageBox.Show("Pincode should not contain a digit repeated more than 3
times", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            TextBox14.Focus()
            Return False
        End If

        Return True
    End Function

    Private Sub ClearInputs()
        ' Clear all textboxes
        TextBox1.Text = "SG"
        TextBox2.Text = ""
        TextBox3.Text = ""
        DateTimePicker1.Value = DateTime.Today ' Clear DateTimePicker1
        TextBox5.Text = "+91"
        TextBox6.Text = ""
        TextBox7.Text = ""
        TextBox9.Text = ""
        TextBox10.Text = ""
        TextBox11.Text = ""
        TextBox12.Text = ""
        TextBox14.Text = ""

        ' Reset to default ComboBox1 (PhotoID)
        ComboBox1.SelectedIndex = 0
        ' Clear ComboBox2 (State)
        ComboBox2.SelectedIndex = -1
        'Clear Designation ComboBox
        ComboBox3.SelectedIndex = 0
    End Sub

    Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs) Handles
TextBox1.TextChanged

 ' Generate the next employee ID and display it in TextBox1
        TextBox1.Text = GenerateNextEmpIDFromDatabase()

        ' Disable TextBox1 to prevent editing
        TextBox1.Enabled = False
    End Sub

    Private Sub TextBox2_TextChanged(sender As Object, e As EventArgs) Handles
TextBox2.TextChanged
        If TextBox2.Text.Length > 0 Then
            ' Get the current selection start position
            Dim selectionStart As Integer = TextBox2.SelectionStart
            ' Capitalize the first letter
            TextBox2.Text = TextBox2.Text.Substring(0, 1).ToUpper() +
TextBox2.Text.Substring(1)
            ' Restore the selection start position
            TextBox2.SelectionStart = selectionStart
        End If
    End Sub
    Private Sub TextBox3_TextChanged(sender As Object, e As EventArgs) Handles
TextBox3.TextChanged
        If TextBox2.Text.Length > 0 Then
            ' Get the current selection start position
            Dim selectionStart As Integer = TextBox3.SelectionStart
            ' Capitalize the first letter
            TextBox3.Text = TextBox3.Text.Substring(0, 1).ToUpper() +
TextBox3.Text.Substring(1)
            ' Restore the selection start position
            TextBox3.SelectionStart = selectionStart
        End If

    End Sub
    Private Sub TextBox10_TextChanged(sender As Object, e As EventArgs) Handles
TextBox10.TextChanged

        If TextBox10.Text.Length > 0 Then
            ' Get the current selection start position
            Dim selectionStart As Integer = TextBox10.SelectionStart
            ' Capitalize the first letter
            TextBox10.Text = TextBox10.Text.Substring(0, 1).ToUpper() +
TextBox10.Text.Substring(1)
            ' Restore the selection start position
            TextBox10.SelectionStart = selectionStart
        End If
    End Sub

    Private Sub TextBox9_TextChanged(sender As Object, e As EventArgs) Handles
TextBox9.TextChanged
        If TextBox9.Text.Length > 0 Then
            ' Get the current selection start position
            Dim selectionStart As Integer = TextBox9.SelectionStart
            ' Capitalize the first letter
            TextBox9.Text = TextBox9.Text.Substring(0, 1).ToUpper() +
TextBox9.Text.Substring(1)
```

```
            ' Restore the selection start position
            TextBox9.SelectionStart = selectionStart
        End If
    End Sub
    Private Sub TextBox12_TextChanged(sender As Object, e As EventArgs) Handles
TextBox12.TextChanged

        If TextBox12.Text.Length > 0 Then
            ' Get the current selection start position
            Dim selectionStart As Integer = TextBox12.SelectionStart
            ' Capitalize the first letter
            TextBox12.Text = TextBox12.Text.Substring(0, 1).ToUpper() +
TextBox12.Text.Substring(1)

            ' Restore the selection start position
            TextBox12.SelectionStart = selectionStart
        End If
    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles
Button3.Click
        ' Clear all textboxes and reset controls
        TextBox1.Text = "SG"  ' Employee ID set to SG
        TextBox2.Text = ""  ' First Name
        TextBox3.Text = ""  ' Last Name
        DateTimePicker1.Value = DateTime.Today  ' DOB to current date
        TextBox5.Text = "+91" ' Phone number back to +91
        TextBox6.Text = ""  ' Alternate phone number
        TextBox7.Text = ""  ' Email ID
        TextBox9.Text = ""  ' House no/name
        TextBox10.Text = "" ' Street
        TextBox11.Text = "" ' City
        TextBox12.Text = "" ' District
        TextBox14.Text = "" ' Pincode
        ComboBox1.SelectedIndex = 0  ' Reset ComboBox1 (PhotoID) to default
        ComboBox2.SelectedIndex = -1 ' Clear ComboBox2 (State)
        'Clear Designation ComboBox
        ComboBox3.SelectedIndex = 0
    End Sub

    Private Sub TextBox5_TextChanged(sender As Object, e As EventArgs) Handles
TextBox5.TextChanged

    End Sub

    Private Sub Label17_Click(sender As Object, e As EventArgs) Handles
Label17.Click

        ' Exit the application

        Application.Exit()
    End Sub
End Class
```

Site Assignment Page

```
Imports System.Globalization
Imports System.Text.RegularExpressions
Imports System.Data
Imports System.Data.SqlClient

Public Class Form6

    ' Database connection string
    Private connectionString As String = "Data Source=DESKTOP-O16HO1G\SQLEXPRESS;Initial
Catalog=Security Service Management System;Integrated Security=True;"
    Private Sub Button1_Click(sender As Object, e As EventArgs) Handles Button1.Click

        ' Format for EMP ID
        Dim regexEmpID As New Regex("^SG[1-9]\d{0,4}$")
        ' Format for Site ID
        Dim regexSITEID As New Regex("^S[1-9]\d{0,4}$")

        ' Initialising for Start date and End date
        Dim startDate As Date
        Dim endDate As Date

        ' Employee ID Presence Check
        If String.IsNullOrWhiteSpace(TextBox1.Text) Then

          MessageBox.Show("Employee ID cannot be blank", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
          TextBox1.Focus()
          Exit Sub
        ElseIf Not regexEmpID.IsMatch(TextBox1.Text) Then
          MessageBox.Show("Invalid EMPLOYEE ID", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
          TextBox1.Focus()

          Exit Sub
        End If

        ' Site ID Presence Check
        If String.IsNullOrWhiteSpace(TextBox2.Text) Then
          MessageBox.Show("Site ID cannot be blank", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
          TextBox2.Focus()
          Exit Sub
        ElseIf Not regexSITEID.IsMatch(TextBox2.Text) Then
          MessageBox.Show("Invalid SITE ID", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
          TextBox2.Focus()
          Exit Sub
        End If

        ' Check if both start date and end date are selected
        If Not DateTimePicker1.Checked OrElse Not DateTimePicker2.Checked Then
          MessageBox.Show("Both Start date and End date are required", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
          Exit Sub
```

```
        End If

        ' Extract start and end dates from DateTimePicker controls
        startDate = DateTimePicker1.Value.Date
        endDate = DateTimePicker2.Value.Date

        ' Check if Start Date is after End Date or same as End Date
        If startDate >= endDate Then
            MessageBox.Show("Start Date cannot be equal to or after End Date.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If



        ' Define duration
        Dim duration As TimeSpan = endDate - startDate

        ' Check if Start Date is in the past
        If startDate < Date.Today Then
            MessageBox.Show("Start Date should not be in the past.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        ' Check If the duration Is at least 1 month
        If duration.TotalDays < 30 Then
            MessageBox.Show("The duration between Start Date and End Date should be at least 1 month.",
"Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        ' Check if End Date is more than 1 year from the Start Date
        If endDate > startDate.AddYears(1) Then
            MessageBox.Show("End Date should not be more than 1 year from the Start Date.", "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        ' Check if Employee ID exists
        If Not EmployeeExists(TextBox1.Text) Then
```

```
            MessageBox.Show("Employee ID does not exist.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            TextBox1.Focus()
            Exit Sub
        End If

        ' Check if Site ID exists
        If Not SiteExists(TextBox2.Text) Then
            MessageBox.Show("Site ID does not exist.", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            TextBox2.Focus()
            Exit Sub
        End If

        ' Check for overlapping assignments
        If AssignmentOverlap(TextBox1.Text, startDate, endDate) Then
            Exit Sub

        End If

        ' If all validations pass, insert data into the database
        If InsertAssignment(TextBox1.Text, TextBox2.Text, startDate, endDate) Then
            MessageBox.Show("Saved Successfully")
            ' Clear all textboxes and reset DateTimePicker controls
            ClearInputs()
        Else
            MessageBox.Show("Error occurred while saving data.", "Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)

        End If
    End Sub

    ' Function to check if an employee exists
    Private Function EmployeeExists(empId As String) As Boolean
        Dim query As String = "SELECT COUNT(*) FROM Emp_table WHERE empid = @empid"
        Using connection As New SqlConnection(connectionString)
            Using command As New SqlCommand(query, connection)
                connection.Open()
                command.Parameters.AddWithValue("@empid", empId)
                Dim count As Integer = Convert.ToInt32(command.ExecuteScalar())
                Return count > 0
            End Using
        End Using
    End Function

    ' Function to check if a site exists
    Private Function SiteExists(siteId As String) As Boolean
        Dim query As String = "SELECT COUNT(*) FROM Site_table WHERE siteid = @siteid"
        Using connection As New SqlConnection(connectionString)
            Using command As New SqlCommand(query, connection)
                connection.Open()
                command.Parameters.AddWithValue("@siteid", siteId)
                Dim count As Integer = Convert.ToInt32(command.ExecuteScalar())
                Return count > 0
            End Using
        End Using
    End Function

    ' Function to check for overlapping assignments
    Private Function AssignmentOverlap(empId As String, startDate As Date, endDate As Date) As Boolean
        Dim query As String = "SELECT COUNT(*), MAX(end_date) FROM site_assignment_table WHERE
empid = @empid AND ((strt_date <= @endDate AND end_date >= @startDate) OR (strt_date >=
@startDate AND end_date <= @endDate))"
        Using connection As New SqlConnection(connectionString)
            Using command As New SqlCommand(query, connection)
                connection.Open()
                command.Parameters.AddWithValue("@empid", empId)
                command.Parameters.AddWithValue("@startDate", startDate)
                command.Parameters.AddWithValue("@endDate", endDate)

                Dim reader As SqlDataReader = command.ExecuteReader()
                If reader.Read() Then
                    Dim count As Integer = Convert.ToInt32(reader(0))
                    If count > 0 Then
                        Dim overlappingEndDate As Date = Convert.ToDateTime(reader(1))
                        MessageBox.Show("There is already an assignment for this employee within the specified
date range. Overlapping assignment ends on: " & overlappingEndDate.ToShortDateString(), "Validation
Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
                        Return True
                    End If
                End If
            End Using
        End Using
        Return False
    End Function

    ' Function to insert assignment into the database
    Private Function InsertAssignment(empId As String, siteId As String, startDate As Date, endDate As
Date) As Boolean
        Dim query As String = "INSERT INTO site_assignment_table (empid, siteid, strt_date, end_date)
VALUES (@empid, @siteid, @startDate, @endDate)"
        Using connection As New SqlConnection(connectionString)
            Using command As New SqlCommand(query, connection)
                connection.Open()
                command.Parameters.AddWithValue("@empid", empId)
                command.Parameters.AddWithValue("@siteid", siteId)
                command.Parameters.AddWithValue("@startDate", startDate)
                command.Parameters.AddWithValue("@endDate", endDate)
                Return command.ExecuteNonQuery() > 0
            End Using

        End Using
    End Function

    ' Function to clear all input fields
    Private Sub ClearInputs()
        TextBox1.Text = "SG"
        TextBox2.Text = "S"
        DateTimePicker1.Value = Date.Today
        DateTimePicker2.Value = Date.Today
    End Sub


    Private Sub Button2_Click(sender As Object, e As EventArgs) Handles Button2.Click
        ' Close the current form (Form6) and open the first form (Form3) again
        Dim form3 As New Form3()
        form3.Show()
        Me.Close()
    End Sub

    Private Sub Form6_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        ' Set the form's AcceptButton property to Button1
        Me.AcceptButton = Button1

        ' Emp ID to start with "SG"
        TextBox1.Text = "SG"

        ' Site ID to start with "S"
        TextBox2.Text = "S"

    End Sub

    Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click
        ' Reset Employee ID to start with "SG"
        TextBox1.Text = "SG"

        ' Reset Site ID to start with "S"
        TextBox2.Text = "S"

        ' Reset Start and End Date to current date
        DateTimePicker1.Value = Date.Today
        DateTimePicker2.Value = Date.Today
    End Sub

    Private Sub Label6_Click(sender As Object, e As EventArgs) Handles Label6.Click
        ' Exit the application
        Application.Exit()
    End Sub
End Class
```

Payroll Generation Page

```
Imports System.Globalization
Imports System.Text.RegularExpressions
Imports System.Data
Imports System.Data.SqlClient
Public Class Form7

    ' Database connection string
    Private connectionString As String = "Data Source=DESKTOP-O16HO1G\SQLEXPRESS;Initial
Catalog=Security Service Management System;Integrated Security=True;"

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

```
        ' Close the current form (Form7) and open the first form (Form3) again
        Dim form3 As New Form3()
        form3.Show()
        Me.Close()
    End Sub

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

        ' Additional check: Ensure that salary has not already been generated for the same employee for the
same month and year
        Dim empId As String = TextBox1.Text
        Dim month As String = ComboBox1.SelectedItem.ToString()
        Dim year As String = ComboBox2.SelectedItem.ToString()
        Dim designation As String = TextBox3.Text

        'Format for EMP ID
        Dim regexEmpID As New Regex("^SG[1-9]\d{0,4}$")

        ' Employee ID Presence Check
        If String.IsNullOrWhiteSpace(TextBox1.Text) Then
            MessageBox.Show("Employee ID cannot be blank", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            TextBox1.Focus()

            Exit Sub
        ElseIf Not regexEmpID.IsMatch(TextBox1.Text) Then
            MessageBox.Show("Invalid EMPLOYEE ID", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
            TextBox1.Focus()

            Exit Sub
        End If

        ' Check if TextBox2 (total salary) is not empty
        If String.IsNullOrEmpty(TextBox2.Text) Then
            MessageBox.Show("Please calculate the total salary before saving.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        ' Check if the employee ID exists in the Emp_table
        If Not EmployeeExists(TextBox1.Text) Then
            MessageBox.Show("Employee ID not Registered. Please register the Employee.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        ' Check if the employee ID exists in the site_assignment_table
        If Not EmployeeAssigned(TextBox1.Text) Then
            MessageBox.Show("Employee ID is not assigned to any site.", "Validation Error",
MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        If SalaryAlreadyGenerated(empId, month, year) Then
            MessageBox.Show("Salary has already been generated for the same employee for the same month
and year.", "Validation Error", MessageBoxButtons.OK, MessageBoxIcon.Error)
            Exit Sub
        End If

        ' If all validations pass, proceed with saving
        SaveToPaymentTable()
        ' Clear textboxes and reset ComboBoxes
        If MessageBox.Show("Salary Generated and Saved Successfully") Then
            ClearInputs()
        End If

    End Sub

    Private Sub TextBox1_TextChanged(sender As Object, e As EventArgs) Handles TextBox1.TextChanged
        ' Only proceed if the TextBox1 contains text and it is different from the default value "SG"


        If TextBox1.Text.Trim().Length > "SG".Length Then
            ' Validate the employee ID format
            Dim empId As String = TextBox1.Text.Trim()
            Dim regexEmpID As New Regex("^SG[1-9]\d{0,4}$")

            If Not regexEmpID.IsMatch(empId) Then
                MessageBox.Show("Invalid EMPLOYEE ID", "Validation Error", MessageBoxButtons.OK,
MessageBoxIcon.Error)
                TextBox1.Focus()
                TextBox3.Text = "" ' Clear TextBox3 if the ID is invalid
            Else
                ' If the format is valid, proceed to fetch the designation
                Dim designation As String = FetchDesignation(empId)
                If Not String.IsNullOrEmpty(designation) Then
                    ' Set the designation in TextBox3
                    TextBox3.Text = designation
                Else
                    ' If designation is not found, clear TextBox3
                    TextBox3.Text = ""
                End If
            End If
        Else
            ' If TextBox1 is empty or contains the default value "SG", clear TextBox3
            TextBox3.Text = ""
        End If
    End Sub

    Private Sub Form7_Load(sender As Object, e As EventArgs) Handles MyBase.Load

        ' Set the form's AcceptButton property to Button1
        Me.AcceptButton = Button1

        ' Set TextBox2 as read-only
        TextBox2.ReadOnly = True



        ' Emp ID to start with "S"
        TextBox1.Text = "SG"

        ' Disable TextBox3
        TextBox3.ReadOnly = True

        ' Get the current month
        Dim currentMonth As Integer = DateTime.Now.Month

        ' Add the range of months to the ComboBox (current month - 1, current month, current month + 1)
        For month As Integer = currentMonth - 1 To currentMonth + 1
            ' Adjust the month to ensure it falls within the range of 1 to 12
            Dim adjustedMonth As Integer = If(month <= 0, month + 12, If(month > 12, month - 12, month))
```

```
          ' Add the month name to the ComboBox
          ComboBox1.Items.Add(DateTimeFormatInfo.CurrentInfo.GetMonthName(adjustedMonth))
      Next

      ' Set the default selected index to the current month (which is the second item in the list)
      ComboBox1.SelectedIndex = 1

      ' Select Current Year
      Dim currentYear As Integer = DateTime.Now.Year
      ' Add the range of years to the ComboBox (current year - 1, current year, current year + 1)
      For year As Integer = currentYear - 1 To currentYear + 1
          ComboBox2.Items.Add(year.ToString())
      Next
      ' Set the default selected index to the current year
      ComboBox2.SelectedIndex = 1 ' Index 1 corresponds to the current year


      ' Add numbers from 0 to 7 to the leaves combo box
      For i As Integer = 0 To 7
          ComboBox3.Items.Add(i.ToString())
      Next
      ' Set the default selected index as o
      ComboBox3.SelectedIndex = 0 ' Selecting 0 by default



      'Disable manual entry into comboBox
      ComboBox1.DropDownStyle = ComboBoxStyle.DropDownList
      ComboBox2.DropDownStyle = ComboBoxStyle.DropDownList
      ComboBox3.DropDownStyle = ComboBoxStyle.DropDownList

  End Sub

  Private Sub Button3_Click(sender As Object, e As EventArgs) Handles Button3.Click

      'Format for EMP ID
      Dim regexEmpID As New Regex("^SG[1-9]\d{0,4}$")

      ' Define salaries for different positions
      Dim salaries As New Dictionary(Of String, Integer) From
      {
          {"Security Guard", 18000},
          {"Lady Guard", 17000},
          {"Field Officer", 22000},
          {"Supervisor", 30000}
      }

      ' Get selected position from TextBox3
      Dim selectedPosition As String = TextBox3.Text.Trim()

      ' Employee ID Presence Check
      If String.IsNullOrWhiteSpace(TextBox1.Text) Then
          MessageBox.Show("Employee ID cannot be blank", "Validation Error", MessageBoxButtons.OK,
  MessageBoxIcon.Error)
          TextBox1.Focus()
          ' Check if the Employee ID matches the pattern "SG" followed by numbers
      ElseIf Not regexEmpID.IsMatch(TextBox1.Text) Then
          MessageBox.Show("Invalid EMPLOYEE ID", "Validation Error", MessageBoxButtons.OK,
  MessageBoxIcon.Error)
          TextBox1.Focus()

          ' Check if designation is selected from ComboBox4
      ElseIf String.IsNullOrWhiteSpace(selectedPosition) Then
          MessageBox.Show("Employee is Not registered", "Validation Error", MessageBoxButtons.OK,
  MessageBoxIcon.Error)
          TextBox3.Focus()

          ' Check if leaves are selected in ComboBox3
      ElseIf ComboBox3.SelectedIndex >= 0 Then

          ' Get the number of leaves from ComboBox3
          Dim leaves As Integer = ComboBox3.SelectedIndex

          ' Calculate deduction for leaves (assuming 500 Rs per leave)
          Dim leaveDeduction As Integer = leaves * 500

          ' Retrieve the corresponding salary based on the selected position
          Dim salary As Integer = salaries(selectedPosition)

          ' Subtract leave deduction from salary only if leaves are not zero
          If leaves > 0 Then
              salary -= leaveDeduction
          End If

          ' Display the final salary in TextBox2
          TextBox2.Text = "Rs " & salary.ToString()
          TextBox1.Enabled = False
          ComboBox1.Enabled = False
          ComboBox2.Enabled = False
          ComboBox3.Enabled = False

      End If
  End Sub

  Private Sub Button4_Click(sender As Object, e As EventArgs) Handles Button4.Click
      ClearInputs()
  End Sub

  ' Function to clear input fields and reset ComboBoxes
  Private Sub ClearInputs()
      ' Set Employee ID as SG
      TextBox1.Text = "SG"

      ' Set the month ComboBox to the current month
      ComboBox1.SelectedIndex = 1

      ' Set the year ComboBox to the current year
      ComboBox2.SelectedItem = DateTime.Now.Year.ToString()

      TextBox3.Text = ""

      ' Set the number of leaves ComboBox to the first option
      ComboBox3.SelectedIndex = 0

      ' Clear the total salary TextBox
      TextBox2.Text = ""

      ' Enable textboxes and comboboxes
      TextBox1.Enabled = True

      ComboBox1.Enabled = True
      ComboBox2.Enabled = True
      ComboBox3.Enabled = True
  End Sub

  ' Function to check if an employee exists in the Emp_table
  Private Function EmployeeExists(empId As String) As Boolean
      Dim query As String = "SELECT COUNT(*) FROM Emp_table WHERE empid = @empid"
      Using connection As New SqlConnection(connectionString)
          Using command As New SqlCommand(query, connection)
              connection.Open()
              command.Parameters.AddWithValue("@empid", empId)
              Dim count As Integer = Convert.ToInt32(command.ExecuteScalar())
              Return count > 0
          End Using
      End Using
  End Function

  ' Function to check if an employee is assigned to any site in the site_assignment_table
  Private Function EmployeeAssigned(empId As String) As Boolean
      Dim query As String = "SELECT COUNT(*) FROM site_assignment_table WHERE empid =
@empid"
      Using connection As New SqlConnection(connectionString)
          Using command As New SqlCommand(query, connection)
              connection.Open()
              command.Parameters.AddWithValue("@empid", empId)
              Dim count As Integer = Convert.ToInt32(command.ExecuteScalar())
              Return count > 0
          End Using
      End Using
  End Function

  ' Function to save data to the payment_table
  Private Sub SaveToPaymentTable()
      ' Extract data from controls
      Dim empid As String = TextBox1.Text

      Dim month As String = ComboBox1.SelectedItem.ToString()
      Dim year As Integer = Convert.ToInt32(ComboBox2.SelectedItem.ToString())
      Dim designation As String = TextBox3.Text
      Dim leaves As Integer = ComboBox3.SelectedIndex
      Dim totalSal As Decimal = Decimal.Parse(TextBox2.Text.Trim().Replace("Rs ", ""),
CultureInfo.InvariantCulture)
```

```
        ' SQL query to insert data into payment_table
        Dim query As String = "INSERT INTO [dbo].[payment_table] ([empid], [month], [year], [designation], 
[leaves], [total_sal]) VALUES (@empid, @month, @year, @designation, @leaves, @total_sal)"

        ' Create connection and command objects
        Using connection As New SqlConnection(connectionString)
            Using command As New SqlCommand(query, connection)
                ' Add parameters
                command.Parameters.AddWithValue("@empid", empid)
                command.Parameters.AddWithValue("@month", month)
                command.Parameters.AddWithValue("@year", year)
                command.Parameters.AddWithValue("@designation", designation)
                command.Parameters.AddWithValue("@leaves", leaves)
                command.Parameters.AddWithValue("@total_sal", totalSal)

                ' Open connection and execute query
                connection.Open()
                command.ExecuteNonQuery()
            End Using
        End Using
    End Sub

    Private Function SalaryAlreadyGenerated(empId As String, month As String, year As String) As Boolean
        ' Query to check if salary has already been generated for the same employee for the same month and 
year
        Dim query As String = "SELECT COUNT(*) FROM payment_table WHERE empid = @empid AND 
[month] = @month AND [year] = @year"

        Using connection As New SqlConnection(connectionString)
            Using command As New SqlCommand(query, connection)
                connection.Open()
                command.Parameters.AddWithValue("@empid", empId)
                command.Parameters.AddWithValue("@month", month)
                command.Parameters.AddWithValue("@year", year)

                ' Execute the query and get the count of records
                Dim count As Integer = Convert.ToInt32(command.ExecuteScalar())

                ' If count is greater than 0, it means salary has already been generated
                Return count > 0
            End Using
        End Using
    End Function
    Private Function FetchDesignation(empId As String) As String
        Dim query As String = "SELECT designation FROM Emp_table WHERE empid = @empid"
        Dim designation As String = String.Empty

        Using connection As New SqlConnection(connectionString)
            Using command As New SqlCommand(query, connection)
                command.Parameters.AddWithValue("@empid", empId)
                Try
                    connection.Open()
                    Dim result As Object = command.ExecuteScalar()
                    If result IsNot Nothing Then
                        designation = Convert.ToString(result)
                    End If
                Catch ex As Exception
                    MessageBox.Show("Error while fetching designation: " & ex.Message)
                End Try
            End Using
        End Using

        Return designation
    End Function

    Private Sub TextBox3_TextChanged(sender As Object, e As EventArgs) Handles TextBox3.TextChanged

    End Sub

    Private Sub TextBox2_TextChanged(sender As Object, e As EventArgs) Handles TextBox2.TextChanged

    End Sub

    Private Sub Label8_Click(sender As Object, e As EventArgs) Handles Label8.Click
        ' Exit the application
        Application.Exit()
    End Sub
End Class
```

6.2.2 OUTPUT DESIGN:

A quality output is one, which meets the requirements of the end user and presents the information clearly.

In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system's relationship to help user decision-making.

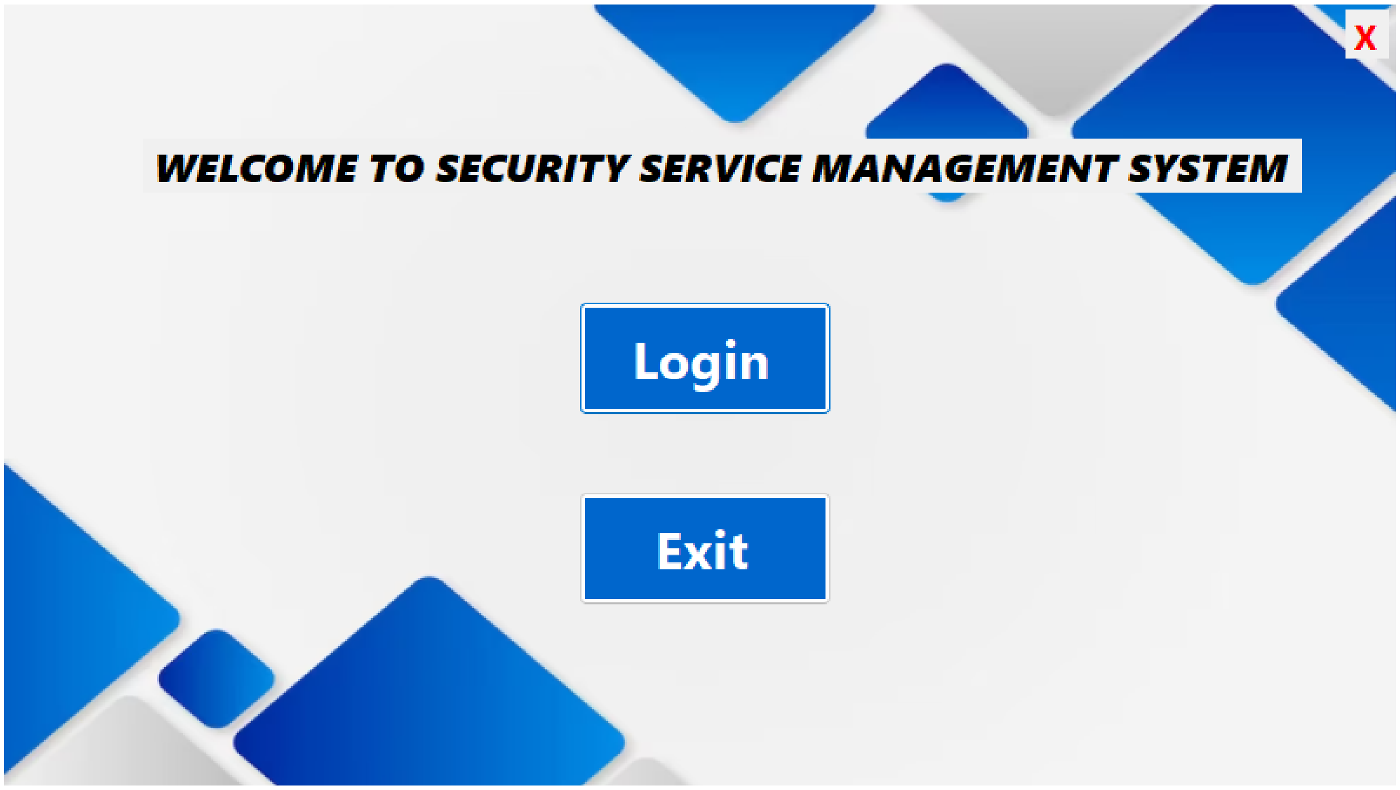
Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should :

Identify the specific output that is needed to meet the requirements. Select methods
for presenting information.
Create document, report, or other formats that contain information produced by the system.

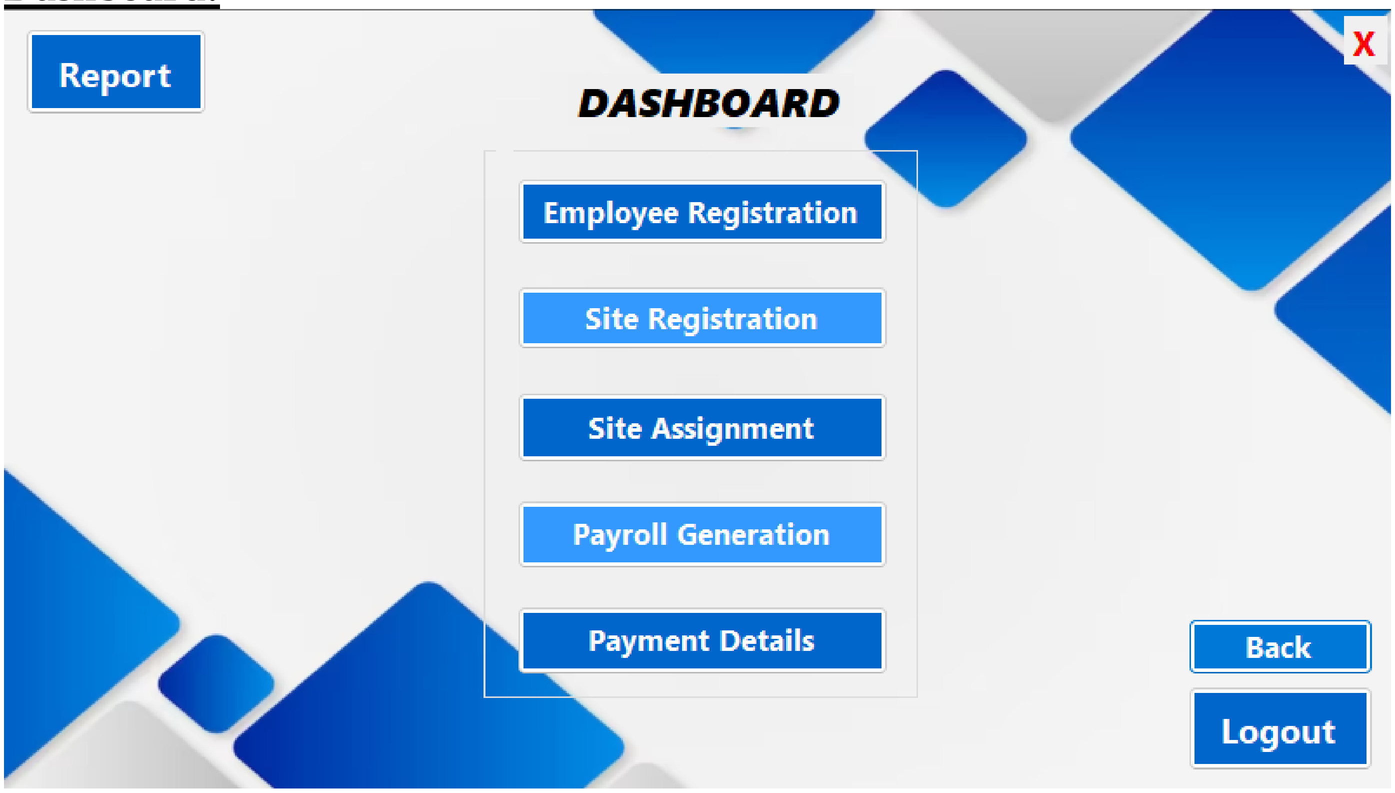
Analysis is the process of extracting the needs of the system and what the system must do to satisfy user's requirements. Object Oriented Analysis is made to develop a series of solution models that describes computer software, which works to satisfy the users. The goal of Object Oriented Analysis is first to understand the domain of the problem and the system responsibilities by understanding hoe the users use or will use the system. This is accomplished by constructing several models of the system. OOA process consists of the following steps:

1. Identify the actors.
2. Develop a simple business process model using UML activity diagram
3. Develop a Use Case.
4. Develop interaction diagrams
5. Identify classes

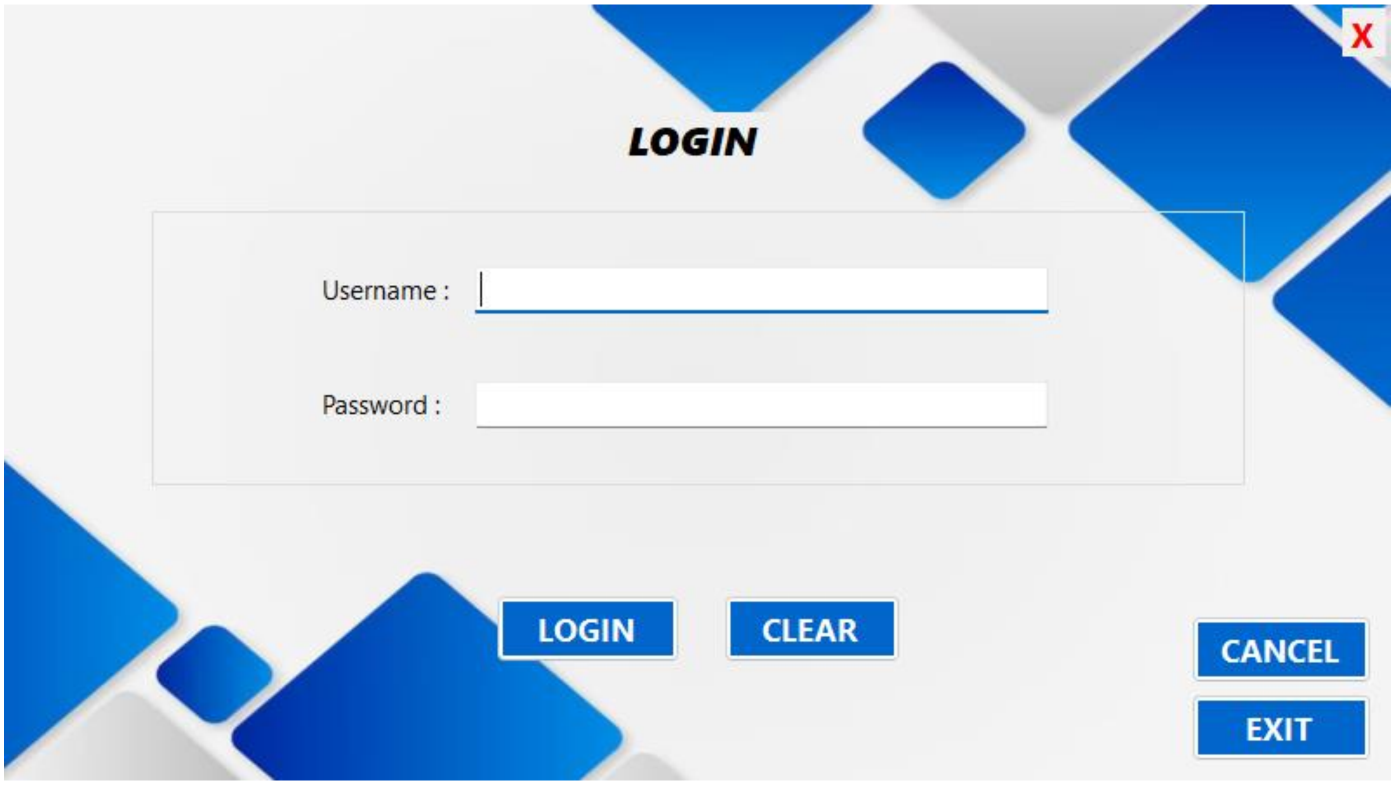
First Page:



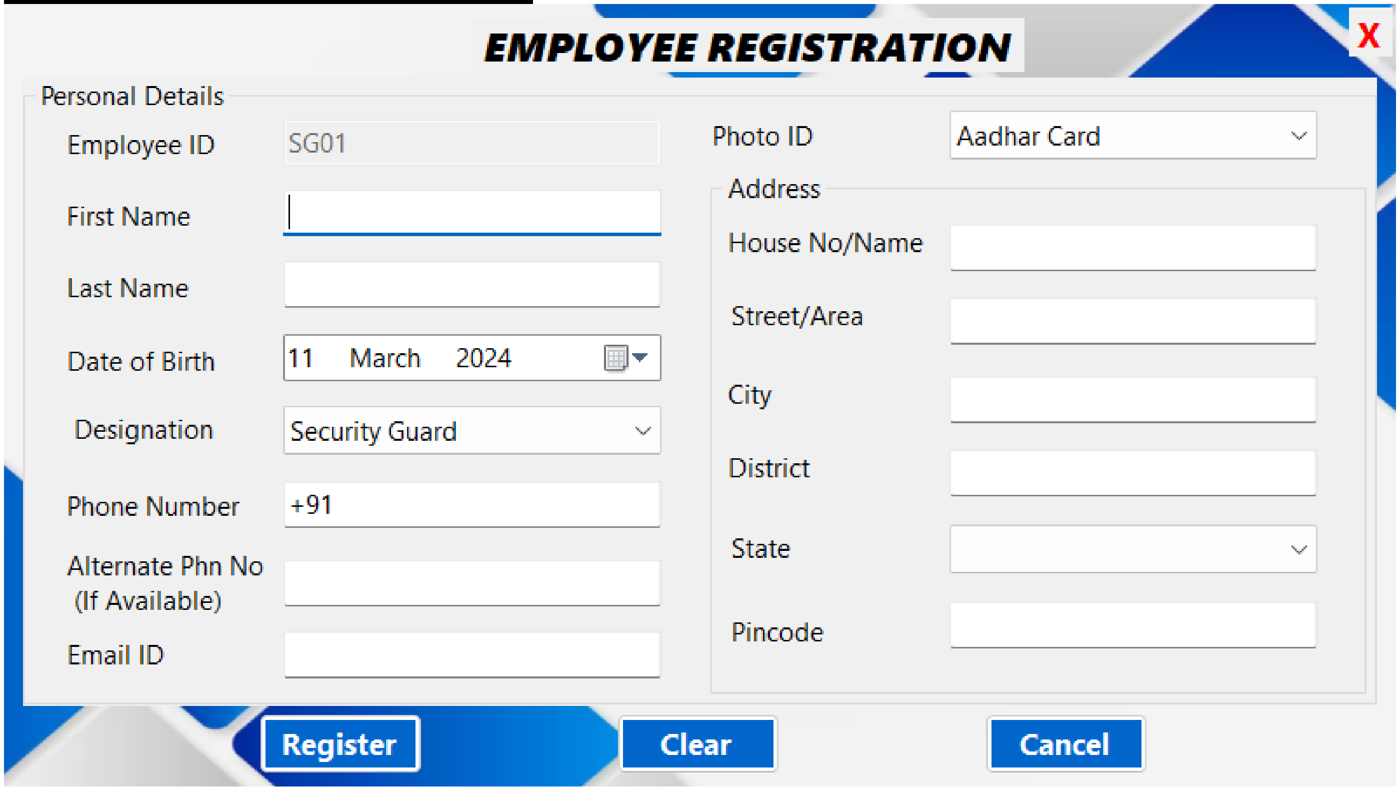
Dashboard:



Admin Login Page:

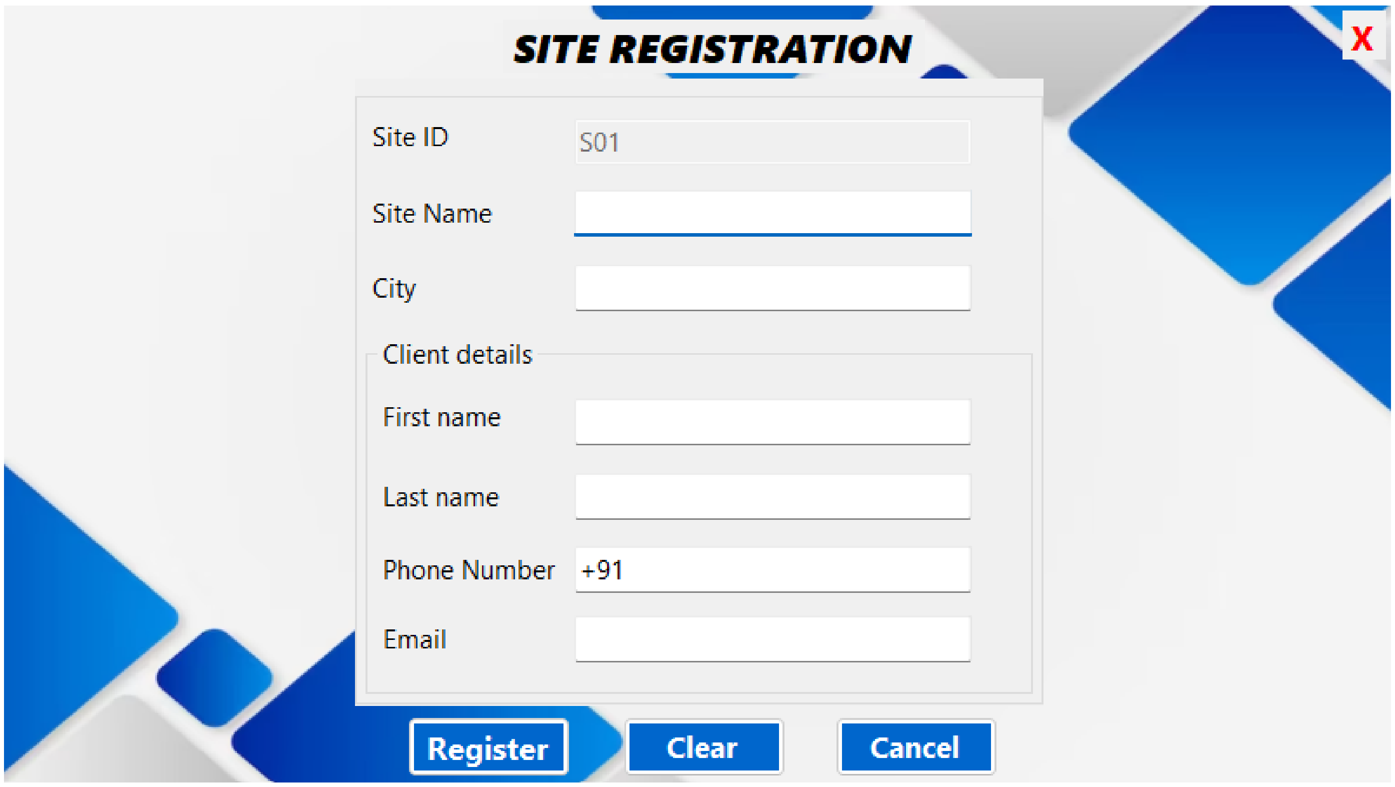


Employee Registration Page:



Employee ID	SG01
First Name	
Last Name	
Date of Birth	11 March 2024
Designation	Security Guard
Phone Number	+91
Alternate Phn No (If Available)	
Email ID	
Photo ID	Aadhar Card
House No/Name	
Street/Area	
City	
District	
State	
Pincode	

Site Registration Page:



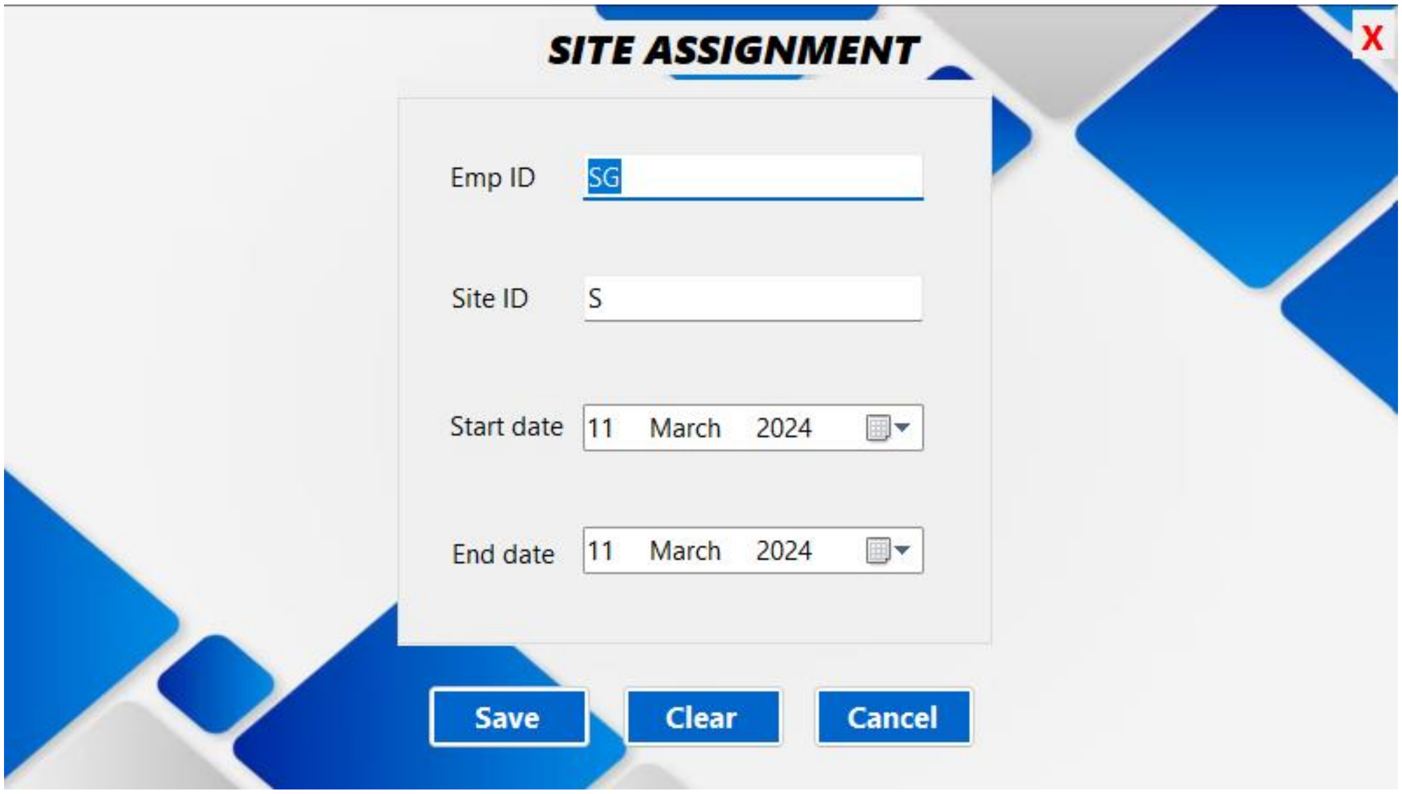
SITE REGISTRATION

Site ID	S01
Site Name	
City	

Client details

First name	
Last name	
Phone Number	+91
Email	

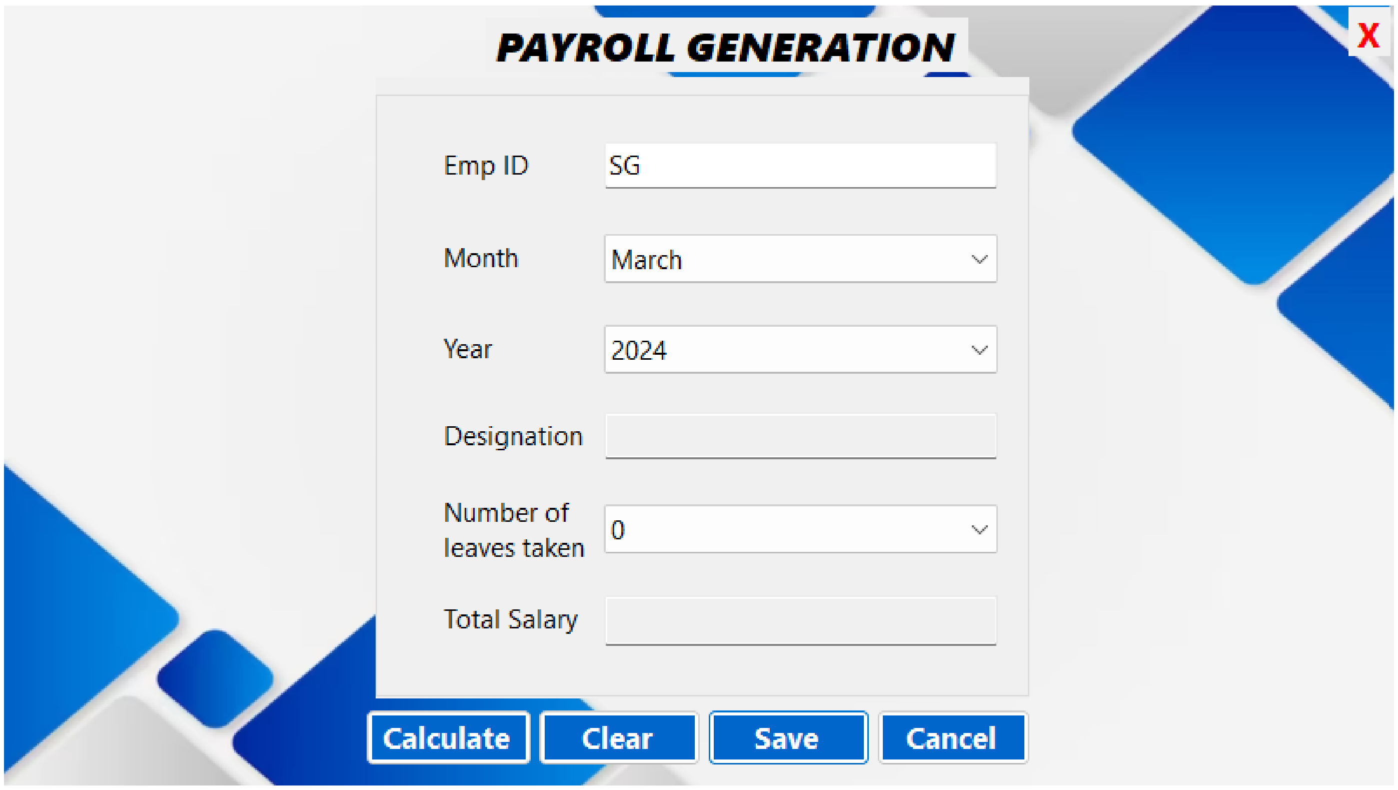
Site Assignment Page:



SITE ASSIGNMENT

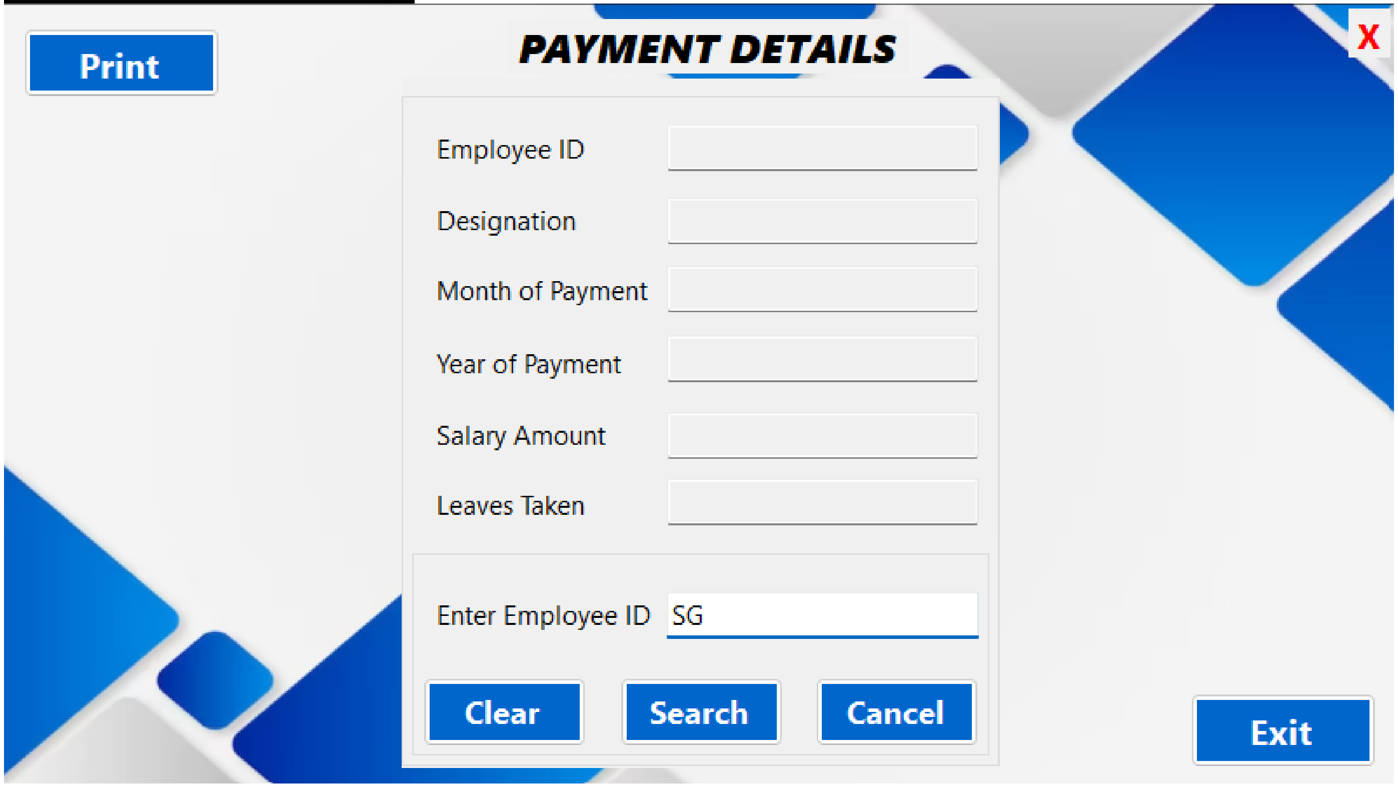
Emp ID	SG
Site ID	S
Start date	11 March 2024
End date	11 March 2024

Payroll Generation Page:



Emp ID	SG
Month	March
Year	2024
Designation	
Number of leaves taken	0
Total Salary	

Payment Details Page:



Employee ID	
Designation	
Month of Payment	
Year of Payment	
Salary Amount	
Leaves Taken	
Enter Employee ID	SG

CHAPTER 7:

TESTING

7. TESTING

Software testing is the crucial element of the software quality assurance and represents the ultimate review of specification, design and coding. Testing represents an interesting anomaly for the software. During earlier definitions and development phases, it was attempted to build software from an abstract concept to tangible information. The testing phase is a very important phase since it is in this phase; we make sure that the system will perform the task without any error. Testing is vital to the success of the system and is being done by classifying it in two ways- System Testing and Program Testing. Program Testing involves checking the syntax and logic of the program. This checking resulted in achieving error free programs.

No matter how carefully a programmer designs and plans application, the programs are sure to have a few bugs in them. Errors in the program immediately stop program execution and display an error message if the errors are syntax errors. After debugging one can identify the limitations of this project and hence corrections are made. During the system development, each source code was tested for its level of correctness. Each form was run a number of times in order to ensure that the details are entered correctly and works properly.

7.1 Test Plan

Testing is the major quality control measure employed during software development. In the project, the first test considered is the unit testing. In this unit testing, each modules of the system are tested separately. This is carried out during programming stage itself. Each module should work satisfactory as regard from the module. After the entire module are checked independently and completed then the integration testing is performed to check whether there is any interface errors. Then those errors are verified and corrected. And also the security test is performed to allow only authorized persons to this system. Finally, the validation testing is performed to validate whether the customer requirements are stratified are not.

7.2 Testing methodologies

1. Unit Testing.
2. Output Testing.
3. Black-Box Test

7.3 Unit Testing:

Unit testing focuses verification efforts on the smallest unit of software design, the module. This is also known as "Module Testing". The modules are tested separately. This testing is carried out during programming stage itself. In these testing steps each module is found to be working satisfactorily as regard to the expected output from the module.

This test can be considered as unit test. This has been carried out after the completion of one complete part. The word validation itself says about the nature of the test. Entire controls in the program have been tested in this manner. The limitations, nature and the boundaries are tested during the test. This test makes the work worthy to be developed further.

7.4 Output Testing:

After performing the validation testing, the next step is output testing of the proposed system since no system could be useful if it does not produce the required output in the specified format. The output generated or displayed by the system under consideration is testing asking the users about the format required by them. Here, the output is considered into two ways: one is on the screen and the other is printed format. The output format on the screen is found to be correct as the format designed according to the user needs. For the hard copy also, the output comes out as specified by the user. Hence output testing doesn't result in any connection in the system.

7.5 Integration testing

The different modules are integrated as per the system design. The different modules are combined into subsystem and then tested to detect errors. The goal of the integration testing is to see if the modules can be integrated properly. It also involves interface testing.

CHAPTER 8:

IMPLEMENTATION

8. IMPLEMENTATION

Implementation includes all those activities that take place to convert from the old system to the new. The old system consists of manual operations, which is operated in a very different manner from the proposed new system. A proper implementation is essential to provide a reliable system to meet the requirements of the organizations. An improper installation may affect the success of the computerized system.

IMPLEMENTATION METHODS

There are several methods for handling the implementation and the consequent conversion from the old to the new computerized system. The most secure method for conversion from the old system to the new system is to run the old and new system in parallel. In this approach, a person may operate in the manual older processing system as well as start operating the new computerized system. This method offers high security, because even if there is a flaw in the computerized system, we can depend upon the manual system. However, the cost for maintaining two systems in parallel is very high. This outweighs its benefits. Another commonly method is a direct cut over from the existing manual system to the computerized system. The change may be within a week or with in a day. There are no parallel activities. However, there is no remedy in case of a problem. This strategy requires careful planning. A working version of the system can also be implemented in one part of the organization and the personnel will be piloting the system and changes can be made as and when required. But this method is less preferable due to the loss of entirety of the system

8.1 IMPLEMENTATION PLAN:

The implementation plan includes a description of all the activities that must occur to implement the new system and to put it into operation. It identifies the personnel responsible for the activities and prepares a time chart for implementing the system. The implementation plan consists of the following steps.

- List all files required for implementation
- Identify all data required to build new files during the implementation.
- List all new documents and procedures that go into the new system

The implementation plan should anticipate possible problems and must be able to deal with them. The usual problems may be missing documents; mixed data formats between current and files, errors in data translation, missing data etc.

POST IMPLEMENTATION REVIEW:

After the system is implemented, a review should be conducted to determine whether the system is meeting expectations and where improvements are needed. System quality, user confidence and operating systems statistics are accessed through such technique event logging, impact evaluation and attitude surveys. The review not only assesses how well the proposed system is designed and implemented, but also is a valuable source of information that can be applied to a critical evaluation of the system.

The reviews are conducted by the operating personals as well as the software developers in order to determine how well the system is working, how it has been accepted and whether adjustments are needed. The review of the system is highly essential to determine the future enhancements required by the system. The system can be considered successful only if information system has met it objectives. The review analyses the opinion of the employees and identifies the attitudes towards the new computerized system. Only when the merits and demerits of the implemented system are known, one can determine what all additional features it requires are. The following are the issues to be considered in the evaluation of the system.

- The change in the cost of operation after the installation of the computerized system.
- The basic change that has been affected after the introduction of the system.
- The improvement in the accuracy of the computations.
- The acceptance of the new system by the staff and the convenience it brought to them.
- The change in the effectiveness caused by the implementation of the new system.

A study of the system has revealed that the employees due to the user friendliness have accepted the system, reduced the number of errors, increased accuracy and decreased cost of operations. The system also pays for efficient and speedy execution of operations compared to the earlier system.

CHAPTER 9:

CONCLUSION AND FUTURE ENHANCEMENT

9. CONCLUSION AND FUTURE ENHANCEMENT

The Security Service Management System project has been successfully completed within the allocated timeframe. Every effort has been made to present the system in a user-friendly manner, ensuring an easy and efficient experience for users. By leveraging modern technologies such as Microsoft Visual Basic 2022 for the front end and Microsoft SQL Server 2019 for the backend, the system offers reliability and scalability in managing security services.

The system effectively addresses the shortcomings of previous systems, enhancing efficiency and productivity in security service management. Through comprehensive testing and refinement, it has been tailored to meet the specific needs of our clients. The user interface is designed with simplicity and accessibility in mind, ensuring that even users with minimal computer proficiency can operate the system effortlessly.

Looking ahead, the Security Service Management System holds promising avenues for enhancement. These include the development of a mobile application to facilitate on-the-go access and management of security services. Additionally, incorporating a feature allowing sites to request additional manpower in cases of shortages would enhance operational flexibility. Implementing a graphical representation of the company's monthly profit or loss would provide valuable insights into its financial performance over time. Furthermore, integrating payroll management capabilities to calculate employee deductions and salaries would streamline administrative processes and ensure accurate compensation. These enhancements aim to elevate the system's functionality, improve operational efficiency, and provide comprehensive tools for decision-making and resource management within the security services domain.

In conclusion, the Security Service Management System has fulfilled its objectives of providing an efficient, user-friendly platform for managing security services. With its robust architecture and intuitive interface, it stands ready to meet the evolving needs of security service providers and clients alike.

CHAPTER 10:

BIBLIOGRAPHY

10. Bibliography

Francesco Balena 2002. Programming Microsoft Visual Basic .NET, US: Microsoft Press.

Anne Boehm 2012. Murach's VB.NET, US: Mike Murach & Associates, Inc.

Bill Sheldon, Billy Hollis, Rob Windsor, David McCarter, Gastón C. Hillar, Todd Herman 2010. Professional Visual Basic 2010 and .NET 4, US: Wiley Publishing, Inc.

Evangelos Petroutsos 2010. Mastering Microsoft Visual Basic 2010, US: Sybex.

Deborah Kurata 2013. Doing Objects in Visual Basic 2005, US: Addison-Wesley.

YouTube links

1.https://www.youtube.com/watch?v=Vx_U7gj4syU

2.https://www.youtube.com/watch?v=-q4HT-od9ds

Websites

1.https://en.wikipedia.org/wiki/SQL_Server_Management_Studio

2.https://en.wikipedia.org/wiki/Visual_Studio

CHAPTER 11:

APPENDICES A

APPENDICES A

Tables Structure

Employee Table:

Field name	Data type	Description	Constraints
empid	varchar(50)	Unique ID for each employee	Primary key
efname	varchar(50)	First Name of employee	NOT NULL
elname	varchar(50)	last Name of employee	NOT NULL
dob	date	Date of birth of Employee	NOT NULL
designation	varchar(50)	Designation of employee	NOT NULL
phn_no	char(13)	Phone number of employee	NOT NULL
alt_phnno	char(13)	Alternate phone number of employee (If available)	ALLOW NULLS
email	varchar(50)	Email id of employee (If available)	ALLOW NULLS
photo_id	varchar(50)	Submitted photo id type	NOT NULL
house_noname	varchar(50)	House name	NOT NULL
street_name	varchar(50)	Street name	NOT NULL
city_name	varchar(50)	City name	NOT NULL
district_name	varchar(50)	District name	NOT NULL
state	varchar(50)	State of Employee	NOT NULL
pincode	int	Pin code of employee	NOT NULL

Site Table:

Field name	Data type	Description	Constraints
siteid	varchar(50)	Unique ID for each site	Primary key
site_name	varchar(50)	name of site	NOT NULL
site_city	varchar(50)	Cityof the site	NOT NULL
cf_name	varchar(50)	First name of client	NOT NULL
cl_name	varchar(50)	Last name of client	NOT NULL
client_phno	Char(13)	Phone number of client	NOT NULL
client_email	varchar(50)	Email id of client	NOT NULL

Site Assignment Table:

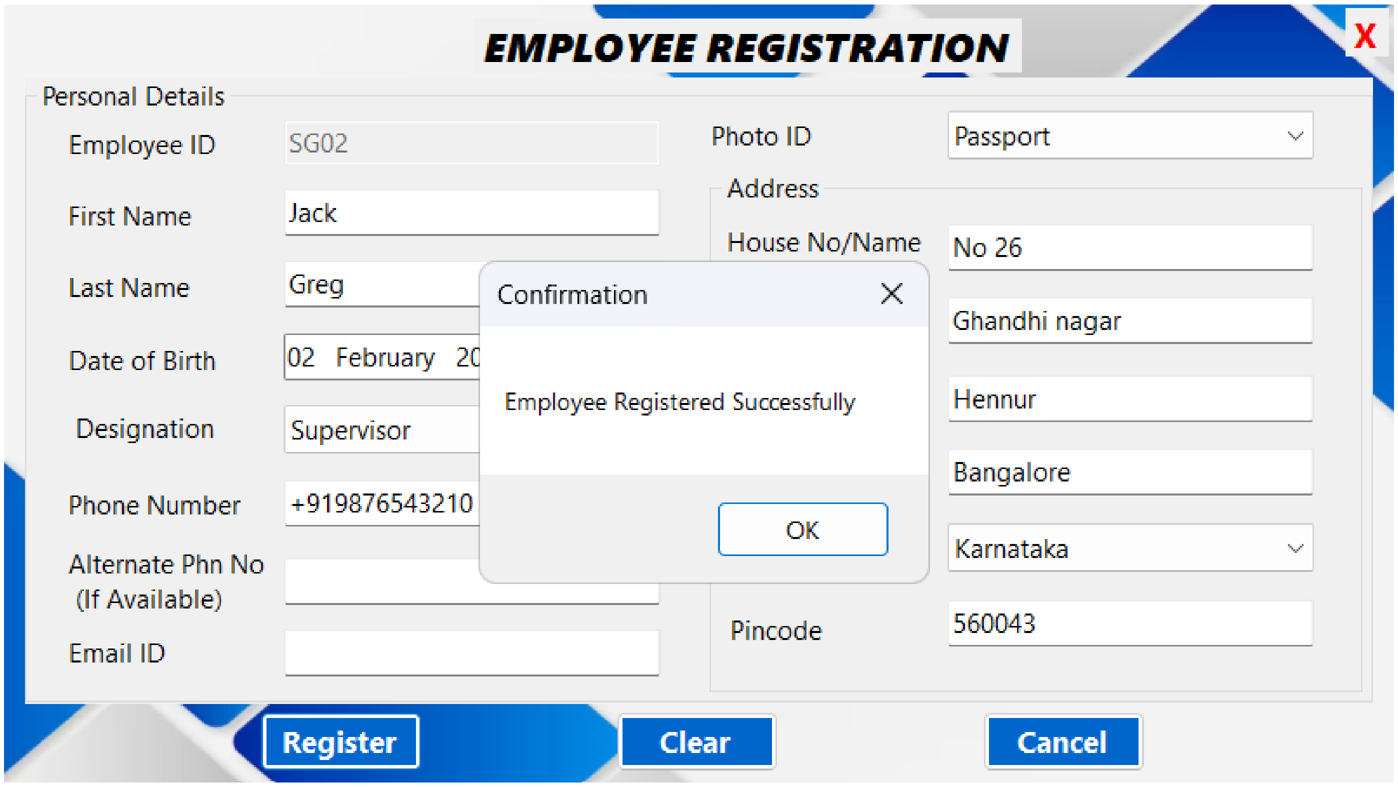
Field name	Data type	Description	Constraints
empid	varchar(50)	Unique ID for each employee	PK,FK
siteid	varchar(50)	Unique ID for each site	FK
strt_date	date	Start date of assignment	PK
end_date	date	End date of assignment	PK

Payment Table:

Field name	Data type	Description	Constraints
empid	varchar(50)	Unique ID for each employee	PK,FK
month	varchar(50)	MONTH OF PAYMENT	PK
year	int	YEAR OF PAYMENT	PK
designation	varchar(50)	DESIGNATION OF EMPLOYEE	NOT NULL
leaves	tinyint	NUMBER OF LEAVES TAKEN	NOT NULL
total_sal	decimal(10,2)	TOTAL SALARY	NOT NULL

CHAPTER 11:

APPENDICES B SCREENSHOTS



EMPLOYEE REGISTRATION

Personal Details

Employee ID: SG02

First Name: Jack

Last Name: Greg

Date of Birth: 02 February 20

Designation: Supervisor

Phone Number: +919876543210

Alternate Phn No (If Available):

Email ID:

Photo ID: Passport

Address

House No/Name: No 26

Ghandhi nagar

Hennur

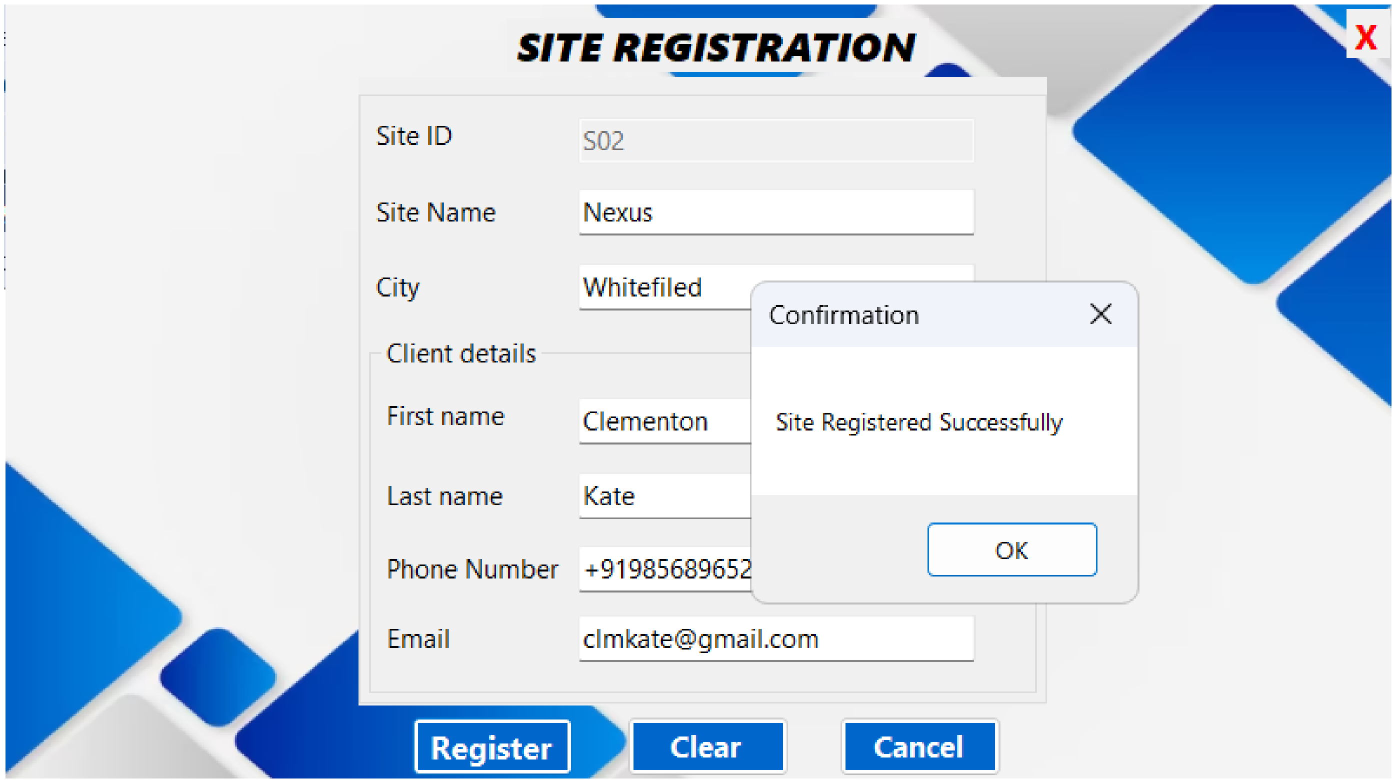
Bangalore

Karnataka

Pincode: 560043

Confirmation

Employee Registered Successfully



SITE REGISTRATION

Site ID: S02

Site Name: Nexus

City: Whitefiled

Client details

First name: Clementon

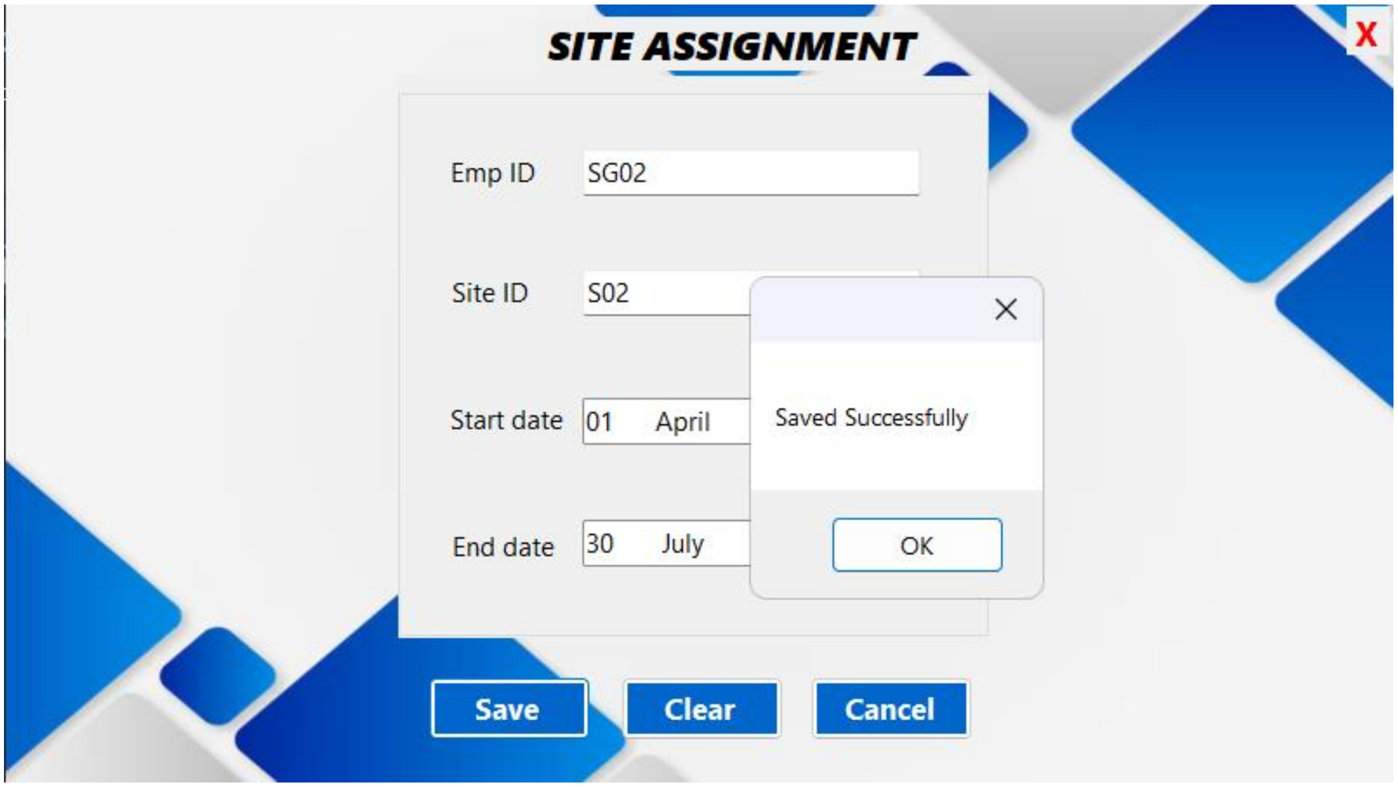
Last name: Kate

Phone Number: +91985689652

Email: clmkate@gmail.com

Confirmation

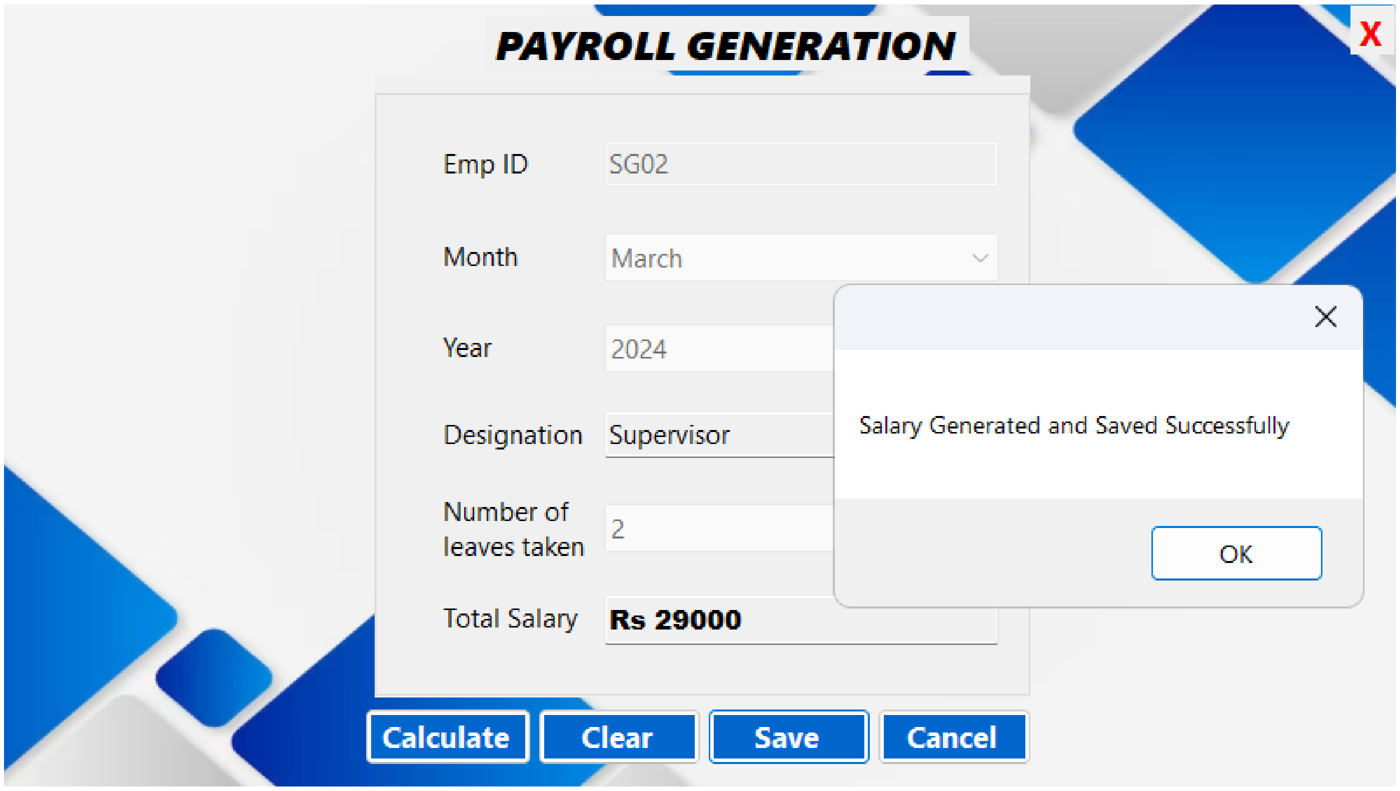
Site Registered Successfully



SITE ASSIGNMENT

Emp ID	SG02
Site ID	S02
Start date	01 April
End date	30 July

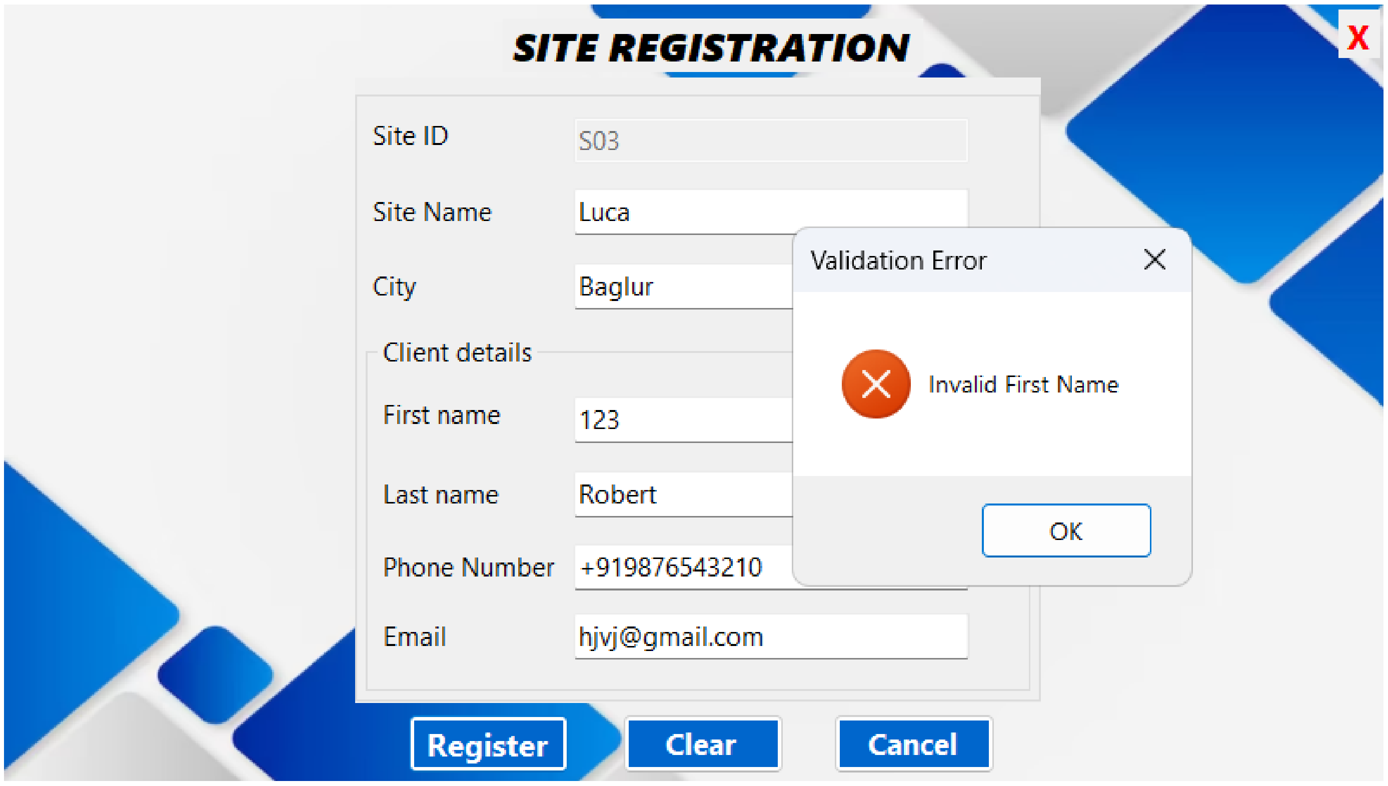
Saved Successfully



PAYROLL GENERATION

Emp ID	SG02
Month	March
Year	2024
Designation	Supervisor
Number of leaves taken	2
Total Salary	Rs 29000

Salary Generated and Saved Successfully



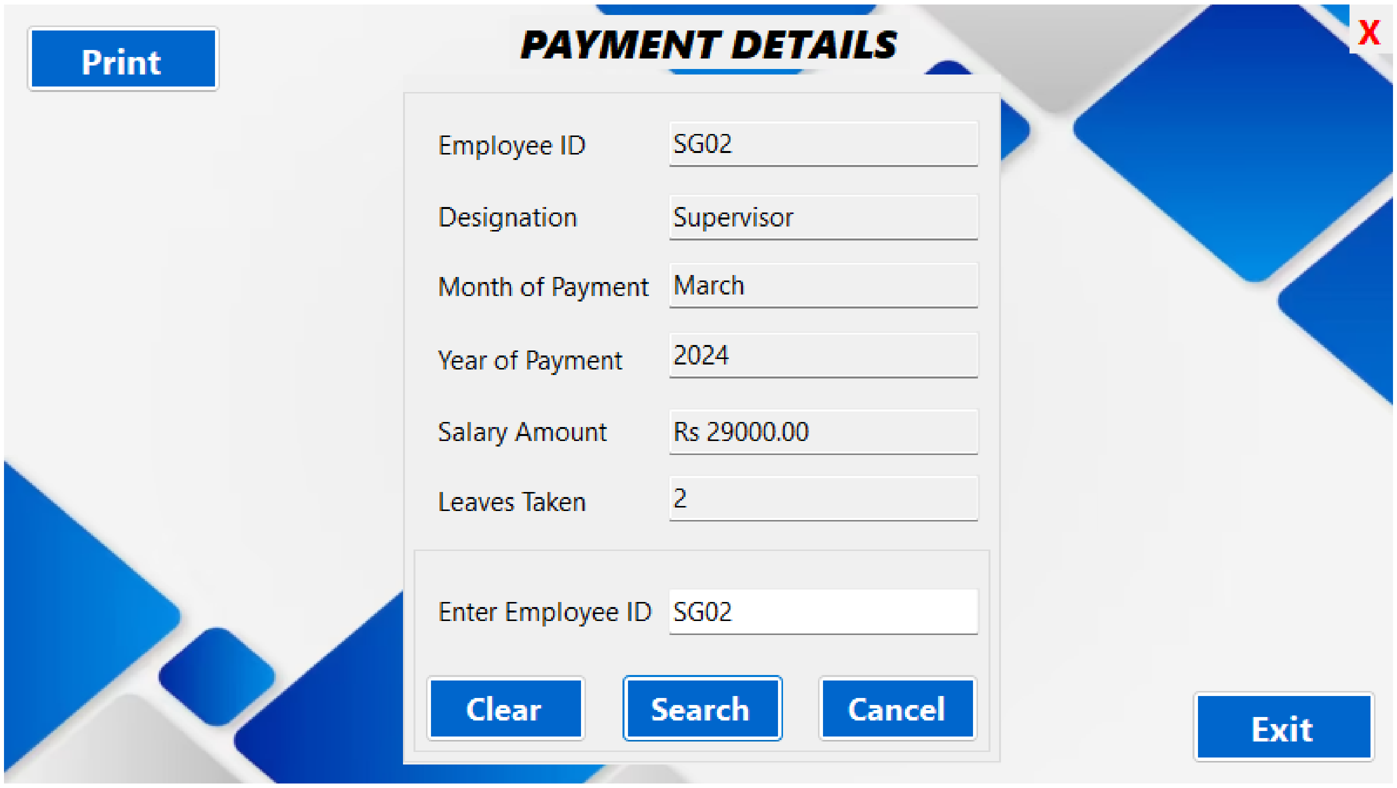
Site ID	S03
Site Name	Luca
City	Baglur

Client details

First name	123
Last name	Robert
Phone Number	+919876543210
Email	hjvj@gmail.com

Validation Error

Invalid First Name



Employee ID	SG02
Designation	Supervisor
Month of Payment	March
Year of Payment	2024
Salary Amount	Rs 29000.00
Leaves Taken	2
Enter Employee ID	SG02

CHAPTER 11:

APPENDICES C
SAMPLE REPORT OF TEST CASES

11. Test cases

Project Name	**Security Service Management System**						
Module Name	**Admin Login**						
Test Reviewed By-	**Prof. Simmi S**						
Requirements ID	**Test Case ID**	**Test Name**	**Test Prerequisites**	**Form**	**Expected Result**	**Actual Result**	**Test Status**
Admin Login	TC_1	Login	Valid user name valid password	Login Form	Successful login	Login Successful	Pass
Admin Login	TC_2	Login Wrong credentials	Invalid Username and password	Login Form	Unsuccessful Login	Invalid Username or Password	Pass
Employee Reg	TC_3	Employee Registration	Valid employee details	Emp Reg Form	Employee Registration	Employee Registered Successfully	Pass
Employee reg	TC_4	Employee Registration With Invalid DOB	DOB in future	Emp Reg Form	Error Message	DOB cannot be in the future	Pass
Site Reg	TC_5	Site Registration With invalid site name	Invalid site name	Site Reg Form	Error Message	Invalid Site name	Pass

Site Reg	TC_6	Site Registration	Valid Site details	Site Reg Form	Successful site registration	Site Registered Successfully	Pass
Site Assignment	TC_7	Site Assignment	Valid emp_id and siteid	Site Assignment Form	Assign successful	Saved Successfully	Pass
Site Assignment	TC_8	Site Assignment with unregistered site	Site_id that is not registered	Site Assignment Form	Error Message	Site is not registered	Pass
Payroll Generation	TC_9	Payroll Generation	Valid emp_id and no of leaves	Payroll generation form	Successful salary generation and saving	Salary generated and saved successfully	Pass
Payroll Generation	TC_10	Payroll Generation with unregistered emp_id	Emp_id which is not registered	Payroll generation form	Error message	Employee is not registered	Pass
Payment Details	TC_11	Payment details searching with wrong emp_id	Unregistered emp_id	Payment Details	Error Message	Employee ID does not exist	Pass
Payment Details	TC_12	Payment details searching with valid emp_id	Valid emp_id	Payment Details	Display the salary details	Salary details is displayed	Pass

Site Assignment	TC_13	Site Assignment with overlapping dates	Overlapping assignment dates	Site Assignment Form	Error message	There is already an assignment for this employee within the specified date range. Overlapping assignment ends on: 30-07-2024	Pass
Site Assignment	TC_14	Site Assignment with end date before start date	Start date which comes after end date	Site Assignment Form	Error Message	Start date cannot be equal to or after end date	Pass
Payroll Generation	TC_15	Payroll Generation for already generate employee	Emp_id whose salary is already saved for same month and year	Payroll generation form	Error Message	Salary has already been generated for the same employee for the same month and year.	Pass