**4.1 Technical Documentation**

Table of Contents

[4.1.1 Hardware and Software Requirements 2](#_Toc352959751)

[4.1.1.1Software Requirements 2](#_Toc352959752)

[4.1.1.2 Hardware Requirements 2](#_Toc352959753)

[4.1.2 Data Structures 3](#_Toc352959754)

[4.1.3 Entity Relationship Diagrams 7](#_Toc352959760)

[4.1.4 Input Forms 8](#_Toc352959762)

[4.1.5 Output Forms 13](#_Toc352959769)

[4.1.6 List of Formulae Used 16](#_Toc352959774)

[4.1.7 System Schematics 17](#_Toc352959775)

[System Flowchart 17](#_Toc352959776)

[Data Flow Diagrams 17](#_Toc352959777)

[4.1.8 Code Listing 18](#_Toc352959779)

# 4.1.1 Hardware and Software Requirements

## 4.1.1.1Software Requirements

1. Microsoft Visual Studio 2008: It will be used as the platform to design the application using the Visual C#. The software is not needed after the program has been compiled and the installation package has been prepared.
2. Microsoft SQL Server 2005: It will be used as the Database Management Software to hold the data of the payroll system. It is required in the operating computer.
3. Microsoft Word 2007: It will be used to prepare the Documentation and the User Manual for the system.
4. Microsoft Visio 2007: It will be used to prepare the diagrammatic explanations like Data Flow Diagrams, System Flowcharts.
5. An operating system strong enough to handle the powerful applications. E.g. Windows 7

## 4.1.1.2 Hardware Requirements

1. Pentium IV system with the processor-speed of 1.6 GHz or above to ensure that the system processes quickly.
2. 512 MB or more memory so that there is sufficient memory to load heavy applications like Microsoft SQL Server 2005.
3. Hard Disk Drive with minimum capacity of 40 GB or more to ensure that there is sufficient space to install Windows 7, Microsoft SQL Server 2005 and other essential software could be used in the company.

# 4.1.2 Data Structures

There are five tables in my system which stores the data. They are thoroughly normalized and designed in such a way to increase the efficiency of the system and remove data redundancy. The data structures and the table schema of the system are presented below.

### Login Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Field Size (bytes) | Example | Remarks |
| Primary Key | Id | Integer | 4 | 1 | Auto increment |
|  | Username | String | 50 | Capmark | Null Value Not Allowed |
|  | Password | String | 15 | Abcd | Null Value Not Allowed |

Login Table stores the login credentials for the system. It stores the information about the users who can access the system. When user logs in to the system, the credentials is verified in this table.

### Employee Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Field Size (bytes) | Example | Remarks |
| Primary Key | Id | Integer | 4 | 241 | Auto increment |
|  | Fname | String | 50 | Karuna | Null Value Not Allowed |
|  | Lname | String | 50 | Budhathoki | Null Value Not Allowed |
|  | Middlename | String | 50 | Kumari |  |
|  | Dateofbirth | Date/Time | 8 | 05/02/1995 00:00:00 AM | Null Value Not Allowed |
|  | Designation | String | 50 | Manager | Null Value Not Allowed |
|  | Address | String | 50 | Baneshwore | Null Value Not Allowed |
|  | Qualification | String | 50 | A Levels |  |
|  | Contactdetails | String | 50 | 9779849113438 | Null Value Not Allowed |
|  | Active | Boolean | 1 | True | Null Value Not Allowed |
|  | Deleted | Boolean | 1 | False | Null Value Not Allowed |

The Employee table stores the details about the staffs. Id is used as the primary key and it is increased automatically. Other details are the general details of staffs needed for the organization as their personal records.

### Attendance Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Field Size (bytes) | Example | Remarks |
| Primary Key | Id | Int | 4 | 1 | Auto Increment |
| Foreign Key | Employee\_id | Int | 4 | 24 | Null Value Not Allowed, Retrieved from employee table |
|  | Attendance\_date | Date/time | 8 | 04/04/1995 00:00:00 AM | Null Value Not Allowed |
|  | Present | Boolean | 1 | True | Null Value Not Allowed |

The attendance table stores the daily attendance. Id is the primary key of this table which increments every time a new record is added. Employee\_id is retrieved from employee table for whose attendance is to me marked. Attendance\_date is the date for which attendance is marked. Present stores the attendance state. True stands for “Present” and False stands for “Absent”.

### Salary Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Field Size (bytes) | Example | Remarks |
| Primary Key | Id | Integer | 4 | 29 | Auto-increment |
| Foreign Key | Employee\_id | Integer | 4 | 1 | Null Value Not Allowed, Retrieved from employee table |
|  | Basic\_salary | Float | 8 | 20000.00 | Null Value Not Allowed |
|  | Allowance | Float | 8 | 12 | Null Value Not Allowed, Value must be less than 100 |
|  | Insurance | Float | 8 | 10 | Null Value Not Allowed, Value must be less than 100 |

Salary table store the parameter for the monthly salary calculation. Id is the primary key of the table that increases every time a new record is added to the table. Employee\_id is the id for the employee whose salary details are going to be stored. It is the foreign key of the table as it is retrieved from the employee table. Basic\_salary stores the per day salary of the employee. Allowance stores the percentage of allowance and Insurance stores the percentage of insurance.

### Monthly Salary Table

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Key | Field Name | Data Type | Field Size (bytes) | Example | Remarks |
| Primary Key | Id | Integer | 4 | 29 | Auto-increment |
| Foreign Key | Employee\_id | Integer | 4 | 1 | Null Value Not Allowed, Retrieved from employee table |
|  | Month | Datetime | 8 | 2/28/2013 12:00:00 AM | Null Value Not Allowed |
|  | monthlySalary | Float | 8 | 20000.00 | Null Value Not Allowed |
|  | Allowance | Float | 8 | 1200.00 | Null Value Not Allowed |
|  | Insurance | Float | 8 | 1000.00 | Null Value Not Allowed |
|  | netSalary | Float | 8 | 20200.00 | Null Value Not Allowed |

Monthly Salary tables stores the monthly salary of each employee for every month. Id is the primary key of the table and is increased automatically every time a new record is added. Employee\_id is the id for the employee whose salary details are going to be stored. It is the foreign key of the table as it is retrieved from the employee table. Month is the month for which the salary is stored. MonthlySalary is the salary of the employee for that month. Allowance and Insurance stores the calculated value for that employee for that month. netSalary stores the final calculated salary for the employee for that month.

# 4.1.3 Entity Relationship Diagrams



## Relationships:

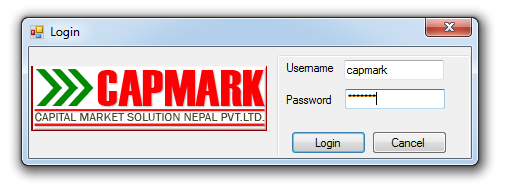
Employee and Attendance : One to Many

Employee and Salary: One to One

Employee and Monthly Salary: One to Many

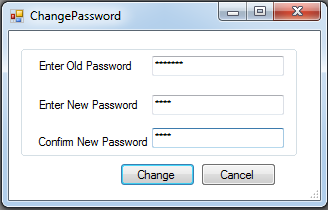
# 4.1.4 Input Forms

### Login Form



* Username is input in the username textbox.
* Password is input in the password textbox.
* Fields cannot be left blank. Error is shown in case any of the field is left blank
* The username and password must match with that in the database.
* The login button directs the system to the main window after the authentication of username and password.
* The close button and cancel button closes the form and exits the application

### Change Password Form



Enter the corresponding values for labels in their field boxes

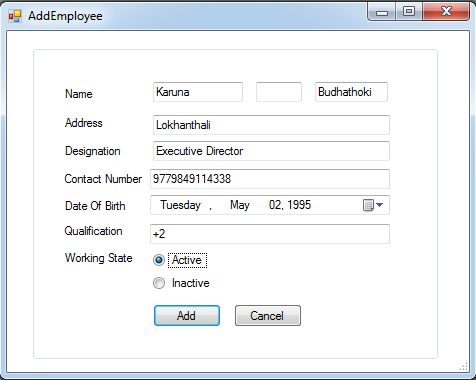
Click this button to update the new password

Click this button to cancel and exit the form

* The old password must match with that in the database.
* The new password and the confirmed password must match.
* The fields cannot be left empty.
* Corresponding errors will be displayed in each case.

### Add Employee Form

Enter the corresponding values for labels in their field boxes. For date, select the date from the date picker. Choose one of the options for the working state.

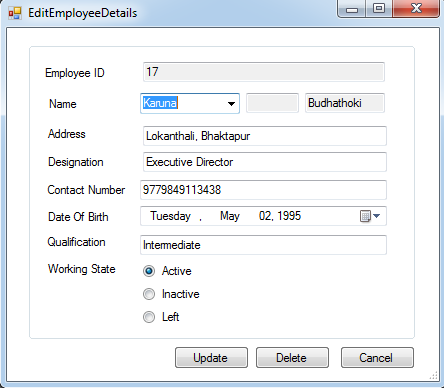


Click this button to cancel and exit the form.

Click this button to add new employee record to the database

* The fields cannot be left empty. Errors will be shown if they are left empty.
* The phone number box does not allow other characters than hyphen (-) and numbers.

### Edit Employee Form



Select an employee for the combobox. To edit, enter the corresponding values for labels in their field boxes. For date, select the date from the date picker. Choose one of the options for the working state.

Click this button to cancel and exit the form

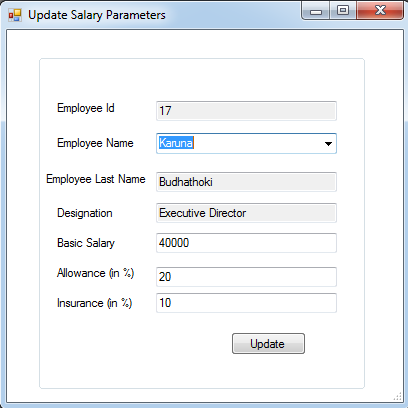
Click this button to delete the employee from the database.

Click this button to update the detail for the selected employee

### 

* The fields Employee ID, middle name and last name cannot be edited.
* The phone number box does not allow other characters than hyphen (-) and numbers.

### Update Salary Parameters Form



Select an employee for the combobox. To edit, enter the corresponding values for labels in their field boxes.

Click this button to update the salary parameter for the selected employee in the database

### 

* The fields Employee ID, last name and designation cannot be edited.
* The basic salary, allowance, allows only numbers to be input.
* The allowance and insurance filed do not allow values more than 100

### Daily Attendance



* Select a date from the date time picker for which to mark the attendance.
* Select the employee from the employee name combo box for whom to mark the attendance.
* Select attendance form the attendance combo box (present or false).
* Click “Done” button to add the attendance d details about the selected employee to the database.
* The date selected cannot exceed the current date.
* The fields Employee Id and Designation cannot be edited.
* An option must be selected for Attendance.
* Attendance for the same employee for the same day cannot be marked more than once.

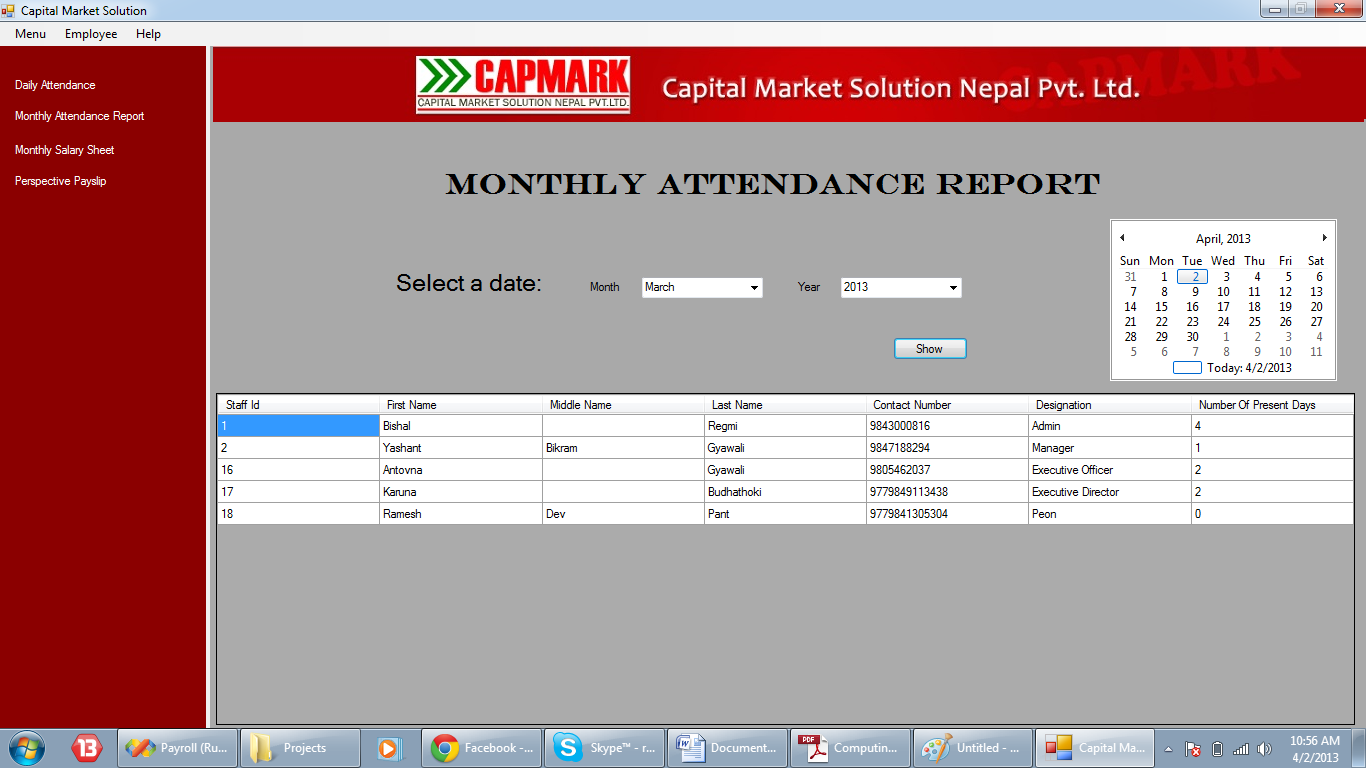
# 4.1.5 Output Forms

### View Employee Report



The report shows the employee details about all the employees in a data grid view.

### Monthly Attendance Report User Control



* Select month and year for which the report is to display from the respective combo boxes.
* On clicking the “Show” button, the attendance report of all the employees for the selected month is shown on the data grid view.

### Monthly Salary Sheet User Control



* Select month and year for which the report is to display from the respective combo boxes.
* On clicking the “Show” button, the monthly salary sheet of all the employees for the selected month is shown on the data grid view.

### Perspective PaySlips User Control



Select month and year for which the payslip is to display from the respective combo boxes.

Click this button to show the salary, allowance, insurance and net salary of the employee on their respective field boxes.

Select employee for whom the payslip is to display from the respective combo boxes.

# 4.1.6 List of Formulae Used

Some of the data to be stored in the database were calculated using the user input values. The formulae for those operations are listed below:

1. To calculate the number of present days for a selected employee for the selected month, the number of records in database was counted with parameters as specific employee id and a range of date and attendance state as True. For example, to get the count of present days for an employee with id “1” for the month of March 2013, query used was,

“string attendanceQuery = "select \* from attendance where employee\_id = 1 and attendance\_date between 2013/3/1 00:00:00.000' and '2013/3/31 00:00:00.000' and present = 'True'";”

1. To calculate the number of monthly salary for an employee formula used was:  
   Monthly Salary = Basic Salary × Number of Present Days
2. To calculate the monthly allowance for an employee for a specific month, formula used was:  
   Allowance = Allowance parameter (% value) × Monthly Salary
3. To calculate the monthly insurance for an employee for a specific month, formula used was:  
   Insurance = Insurance (% value) × Monthly Salary
4. To calculate the Net Salary of the employee for the month, formula used was:  
   Net Salary = Monthly Salary + Allowance – Insurance

# 4.1.7 System Schematics

## System Flowchart



## Data Flow Diagrams

## 

The data is retrieved from the employee table. Their corresponding daily attendance is recorded. A monthly attendance report is prepared. On the basis of monthly attendance report prepared, salary for all the employees is calculated. The calculation is shown in Monthly Salary Sheet and Employee Payslips.

# 4.1.8 Code Listing

### Login Form

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Configuration**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class LoginForm **:** Form

**{**

**public** LoginForm**()**

**{**

InitializeComponent**();**

**}**

/// <summary>

/// This function intends to verify whether the Username and Password Textboxes are left empty or not

/// </summary>

///

/// <returns>values as boolean whether the fields are validated or not</returns>

**private** bool validate**()**

**{**

**if** **(**username\_txtbox**.**Text **==** **null** **||** username\_txtbox**.**Text **==** ""**)**

**{**

alert**.**Text **=** "Username Required"**;**

**return** **false;**

**}**

**if** **(**password\_txtbox**.**Text **==** **null** **||** password\_txtbox**.**Text **==** ""**)**

**{**

alert**.**Text **=** "Password Required"**;**

**return** **false;**

**}**

**return** **true;**

**}**

**private** void cancel\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Close**();**

**}**

**private** void login\_button\_Click\_1**(object** sender**,** EventArgs e**)**

**{**

**if** **(**validate**())**

**{**

checklogin login **=** **new** checklogin**();**

bool isSuccessful **=** login**.**CheckLoginCredential**(**username\_txtbox**.**Text**,** password\_txtbox**.**Text**);**

// sends the values from the username and passwords textboxes to the function checklogin and gets boolean value as verification

**if** **(**isSuccessful**)**

**{**

alert**.**Text **=** ""**;**

UserUtils**.**USERNAME **=** username\_txtbox**.**Text**;**

//Stores the current values in Username Textbox to the USERNAME in the class UsertUtils

MainWindow newfrom **=** **new** MainWindow**();**

newfrom**.**Show**();**

**this.**Hide**();**

**}**

**else**

**{**

alert**.**Text **=** "Wrong credentials"**;**

**}**

**}**

**}**

**}**

**}**

### Usertils Class

This class stores the logged in username as the static username throughout the application run.

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**namespace** Payroll

**{**

class UserUtils

**{**

// Username stores the value of the username with which the user logs in and is static

**public** static string USERNAME**;**

**}**

**}**

### Application Configuration File

The Configuration Manager class enables you to access the connection string for the database.

**<?**xml version**=**"1.0" encoding**=**"utf-8" **?>**

**<**configuration**>**

**<**connectionStrings**>**

**<**add name **=**"payroll\_conn" connectionString **=**"Data Source=BISHALREGMI-HP;Initial Catalog=payroll\_db;User ID=sa;Password=MySQLDatabase " providerName **=**"System.Data.SqlClient"**>**

**</**add**>**

**</**connectionStrings**>**

**</**configuration**>**

### DbConnectionManager Class

This class is used to create, gain and remove connection with the database.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Configuration**;**

**using** System**.**Data**.**SqlClient**;**

**using** System**.**Data**;**

**namespace** Payroll

**{**

class DbConnectionManager

**{**

/// <summary>

/// creates and opens new connection with the database

/// </summary>

/// <returns>connection state with the database</returns>

**public** SqlConnection connectToDb**()**

**{**

string connectionString **=** ConfigurationManager**.**ConnectionStrings**[**"payroll\_conn"**].**ConnectionString**;**

SqlConnection connection **=** **new** SqlConnection**(**connectionString**);**

**if** **(**connection**.**State **!=** ConnectionState**.**Open**)**

**{**

connection**.**Open**();**

**return** connection**;**

**}**

**else**

**{**

**return** **null;**

**}**

**}**

/// <summary>

/// Closes the connection with the database

/// </summary>

/// <param name="conn">SQL Connection String</param>

**public** void diconnectFromDb**(**SqlConnection conn**)**

**{**

**if** **(**conn**.**State **==** ConnectionState**.**Open**)**

**{**

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

**}**

**}**

**}**

**}**

### Checklogin Class

This class is used to check the authentication of the user who logs in to the system. It receives the username and password as from the input forms and checks them with the value in the database for authentication.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Configuration**;**

**using** System**.**Data**.**SqlClient**;**

**using** System**.**Data**;**

**namespace** Payroll

**{**

class checklogin

**{**

/// <summary>

/// This function checks whether the passed username and password exist in the login table or not

/// </summary>

/// <param name="username"> the values for the entered username</param>

/// <param name="password">the values for the entered password </param>

/// <returns> boolean values , true for success and false for failure</returns>

**public** bool CheckLoginCredential**(**string username**,** string password**)**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string query **=** "select \* from login " **+** "where username='" **+** username **+** "' and password='" **+** password **+** "'"**;**

SqlCommand cmd **=** **new** SqlCommand**(**query**);**

cmd**.**Connection **=** conn**;**

SqlDataAdapter da **=** **new** SqlDataAdapter**();**

da**.**SelectCommand **=** cmd**;**

DataSet ds **=** **new** DataSet**();**

da**.**Fill**(**ds**,** "login"**);**

**if** **(**ds**.**Tables**[**"Login"**].**Rows**.**Count **>** 0**)**

**{**

connect**.**diconnectFromDb**(**conn**);**

**return** **true;**

**}**

**else**

**{**

connect**.**diconnectFromDb**(**conn**);**

**return** **false;**

**}**

**}**

**}**

**}**

### Main Window

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** Payroll

**{**

**public** partial class MainWindow **:** Form

**{**

**public** MainWindow**()**

**{**

InitializeComponent**();**

**}**

**private** void MainWindow\_FormClosing**(object** sender**,** FormClosingEventArgs e**)**

**{**

DialogResult dr **=** MessageBox**.**Show**(**"Are You sure that You want to exit?"**,** "Are You Sure?"**,** MessageBoxButtons**.**YesNo**,** MessageBoxIcon**.**Question**);**

**if** **(**dr **==** DialogResult**.**No**)**

**{**

e**.**Cancel **=** **true;**

**}**

//displays a message box to confirm whether the user wants to exit the system or not

**}**

**private** void addEmployeeToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

AddEmployee newform **=** **new** AddEmployee**();**

newform**.**ShowDialog**();**

**}**

**private** void viewEmployeeToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

ViewEmployee newform **=** **new** ViewEmployee**();**

newform**.**Show**();**

**}**

**private** void editEmployeeToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

EditEmployeeDetails newform **=** **new** EditEmployeeDetails**();**

newform**.**ShowDialog**();**

**}**

**private** void changePasswordToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

ChangePassword changepassword **=** **new** ChangePassword**();**

changepassword**.**ShowDialog**();**

**}**

**private** void MainWindow\_FormClosed**(object** sender**,** FormClosedEventArgs e**)**

**{**

Application**.**Exit**();**

**}**

**private** void exitToolStripMenuItem1\_Click**(object** sender**,** EventArgs e**)**

**{**

Application**.**Exit**();**

**}**

**private** void daily\_attendance\_label\_Click**(object** sender**,** EventArgs e**)**

**{**

panel1**.**Controls**.**Clear**();**

DailyAttendance uc **=** **new** DailyAttendance**();**

panel1**.**Controls**.**Add**(**uc**);**

**}**

**private** void monthy\_attendreport\_Click**(object** sender**,** EventArgs e**)**

**{**

panel1**.**Controls**.**Clear**();**

MonthlyAttendanceReport uc **=** **new** MonthlyAttendanceReport**();**

panel1**.**Controls**.**Add**(**uc**);**

**}**

**private** void monthly\_salarysheet\_Click**(object** sender**,** EventArgs e**)**

**{**

panel1**.**Controls**.**Clear**();**

MonthlySalarySheet uc **=** **new** MonthlySalarySheet**();**

panel1**.**Controls**.**Add**(**uc**);**

**}**

**private** void perspective\_payslip\_Click**(object** sender**,** EventArgs e**)**

**{**

panel1**.**Controls**.**Clear**();**

PerspectivePayslip uc **=** **new** PerspectivePayslip**();**

panel1**.**Controls**.**Add**(**uc**);**

**}**

**private** void parameterSettingsToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

SalaryCredentialAssigner newform **=** **new** SalaryCredentialAssigner**();**

newform**.**ShowDialog**();**

**}**

**private** void aboutToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

AboutBox newform **=** **new** AboutBox**();**

newform**.**ShowDialog**();**

**}**

**private** void exitToolStripMenuItem\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Hide**();**

LoginForm loginForm **=** **new** LoginForm**();**

loginForm**.**Show**();**

**}**

**}**

**}**

### Change Password Form

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class ChangePassword **:** Form

**{**

**public** ChangePassword**()**

**{**

InitializeComponent**();**

**}**

**private** void change\_btn\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**oldpassword\_txt**.**Text **==** "" **||** newpassword\_txt**.**Text **==** "" **||** confirmpassword\_txt**.**Text **==** ""**)**

**{**

MessageBox**.**Show**(**"The fields are left empty"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**else**

**{**

//checks whether the new password and the confirm password match or not

**if** **(**newpassword\_txt**.**Text **==** confirmpassword\_txt**.**Text**)**

**{**

//sends the values of the old password and the static username to the function of the class checklogin to confirm the authenticity.

checklogin login **=** **new** checklogin**();**

bool isSuccessful **=** login**.**CheckLoginCredential**(**UserUtils**.**USERNAME**.**ToString**(),** oldpassword\_txt**.**Text**);**

//if authentication is successful then the new password is updated using the query else error is shown in message box

**if** **(**isSuccessful**)**

**{**

string query **=** "update login set password ='" **+** newpassword\_txt**.**Text **+** "' where username = '" **+** UserUtils**.**USERNAME **+** "' "**;**

DbConnectionManager dbConnectionManager **=** **new** DbConnectionManager**();**

SqlConnection connection **=** dbConnectionManager**.**connectToDb**();**

SqlCommand command **=** **new** SqlCommand**(**query**);**

command**.**Connection **=** connection**;**

int result **=** command**.**ExecuteNonQuery**();**

**if** **(**result **>** 0**)**

**{**

MessageBox**.**Show**(**"Password Successfully Changed"**);**

**}**

**else**

**{**

MessageBox**.**Show**(**"Unable to change password"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

dbConnectionManager**.**diconnectFromDb**(**connection**);**

**}**

**}**

**else**

**{**

MessageBox**.**Show**(**"Wrong Password"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**else**

**{**

MessageBox**.**Show**(**"Password do not match"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**}**

**}**

**}**

### Add Employee Form

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class AddEmployee **:** Form

**{**

**public** AddEmployee**()**

**{**

InitializeComponent**();**

**}**

**private** void add\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

**if** **(**fname\_txtbox**.**Text **==** **null** **||** lname\_txtbox**.**Text **==** **null** **||** address\_txtbox**.**Text **==** **null** **||** designation\_txtbox**.**Text **==** **null** **||** contact\_txtbox**.**Text **==** **null** **||** **(**active**.**Checked **==** **false** **&** inactive**.**Checked **==** **false))**

**{**

MessageBox**.**Show**(**"Key fields cannot be left empty"**,** "Validation Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**else**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

//query to update the respective values of the field to the employee table in database

string query **=** "insert into employee(fname,lname,middlename,designation,address,contactdetails,active,deleted,dateofbirth,qualification) values ('" **+** fname\_txtbox**.**Text **+** "','" **+** lname\_txtbox**.**Text **+** "','" **+** mname\_txtbox**.**Text **+** "','" **+** designation\_txtbox**.**Text **+** "','" **+** address\_txtbox**.**Text **+** "','" **+** contact\_txtbox**.**Text **+** "','" **+** active**.**Checked **+** "','False','" **+** dob\_pick**.**Value**.**Date **+** "','" **+** qualification\_txtbox**.**Text **+** "')"**;**

SqlCommand command **=** **new** SqlCommand**(**query**);**

command**.**Connection **=** conn**;**

int result **=** command**.**ExecuteNonQuery**();**

**if** **(**result **>** 0**)**

**{**

// selects the id for the recently added employee

string selectid **=** "select max(id) as id from employee"**;**

command**.**CommandText **=** selectid**;**

command**.**Connection **=** conn**;**

SqlDataAdapter da **=** **new** SqlDataAdapter**();**

da**.**SelectCommand **=** command**;**

DataSet ds **=** **new** DataSet**();**

da**.**Fill**(**ds**,** "employee\_id"**);**

DataRow dr **=** ds**.**Tables**[**"employee\_id"**].**Rows**[**0**];**

string employee\_id **=** dr**[**"id"**].**ToString**();**

// inserts "0" as default salary values for the recently added employee in the salary table

string salary **=** "insert into salary(employee\_id,basic\_salary,allowance, insurance) values(" **+** employee\_id **+** ",0,0,0)"**;**

SqlCommand command1 **=** **new** SqlCommand**(**salary**);**

command1**.**Connection **=** conn**;**

command1**.**ExecuteNonQuery**();**

MessageBox**.**Show**(**"New Employee Added"**);**

connect**.**diconnectFromDb**(**conn**);**

**this.**Close**();**

**}**

**else**

**{**

MessageBox**.**Show**(**"Failed adding New Employee"**);**

**}**

**}**

**}**

**}**

**}**

### Employee Artifact Class

This class contains the list to temporarily store the employee details. Furthermore, this class also contains the functions and queries to update and retrieve the employee details.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Data**;**

**using** System**.**Data**.**SqlClient**;**

**using** System**.**ComponentModel**;**

**namespace** Payroll

**{**

**public** class EmployeeArtifact

**{**

**[**DisplayName**(**"Staff Id"**)]**

**public** int Id **{** get**;** set**;** **}**

**[**DisplayName**(**"First Name"**)]**

**public** string FirstName **{** get**;** set**;** **}**

**[**DisplayName**(**"Last Name"**)]**

**public** string LastName **{** get**;** set**;** **}**

**[**DisplayName**(**"Date Of Birth"**)]**

**public** string DateOfBirth **{** get**;** set**;** **}**

**[**DisplayName**(**"Middle Name"**)]**

**public** string MiddleName **{** get**;** set**;** **}**

**public** string Designation **{** get**;** set**;** **}**

**public** string Address **{** get**;** set**;** **}**

**public** string Qualification **{** get**;** set**;** **}**

**[**DisplayName**(**"Contact Number"**)]**

**public** string ContactDetails **{** get**;** set**;** **}**

**public** bool Active **{** get**;** set**;** **}**

**public** bool Deleted **{** get**;** set**;** **}**

/// <summary>

/// Retrieves all the employee details values from the database and stores them in the EmployeeArtifact List

/// </summary>

/// <returns>list of the employee retrieved from the employee table</returns>

**public** List**<**EmployeeArtifact**>** GetAllEmployee**()**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string employeeQuery **=** "select \* from employee "**;**

SqlCommand cmd1 **=** **new** SqlCommand**(**employeeQuery**);**

cmd1**.**Connection **=** conn**;**

SqlDataAdapter da1 **=** **new** SqlDataAdapter**();**

da1**.**SelectCommand **=** cmd1**;**

DataSet ds1 **=** **new** DataSet**();**

da1**.**Fill**(**ds1**,** "employee"**);**

List**<**EmployeeArtifact**>** empArtifactList **=** **new** List**<**EmployeeArtifact**>();**

//adds every employeedetails in the table to the list

**foreach** **(**DataRow dr **in** ds1**.**Tables**[**"employee"**].**Rows**)**

**{**

EmployeeArtifact temp **=** **new** EmployeeArtifact**();**

temp**.**DateOfBirth **=** dr**[**"dateofbirth"**].**ToString**();**

temp**.**FirstName **=** dr**[**"fname"**].**ToString**();**

temp**.**Address **=** dr**[**"address"**].**ToString**();**

temp**.**Id **=** int**.**Parse**(**dr**[**"id"**].**ToString**());**

temp**.**LastName **=** dr**[**"lname"**].**ToString**();**

temp**.**MiddleName **=** dr**[**"middlename"**].**ToString**();**

temp**.**Designation **=** dr**[**"designation"**].**ToString**();**

temp**.**Qualification **=** dr**[**"qualification"**].**ToString**();**

temp**.**ContactDetails **=** dr**[**"contactdetails"**].**ToString**();**

temp**.**Active **=** bool**.**Parse**(**dr**[**"active"**].**ToString**());**

temp**.**Deleted **=** bool**.**Parse**(**dr**[**"deleted"**].**ToString**());**

empArtifactList**.**Add**(**temp**);**

**}**

**return** empArtifactList**;**

**}**

/// <summary>

/// retrieves all the employees from the employee table whose state is active and updates them to the EmployeeArtifact List

/// </summary>

/// <returns></returns>

**public** List**<**EmployeeArtifact**>** GetActiveEmployee**()**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string employeeQuery **=** "select \* from employee where active = 'True'"**;**

SqlCommand cmd1 **=** **new** SqlCommand**(**employeeQuery**);**

cmd1**.**Connection **=** conn**;**

SqlDataAdapter da1 **=** **new** SqlDataAdapter**();**

da1**.**SelectCommand **=** cmd1**;**

DataSet ds1 **=** **new** DataSet**();**

da1**.**Fill**(**ds1**,** "employee"**);**

List**<**EmployeeArtifact**>** empArtifactList **=** **new** List**<**EmployeeArtifact**>();**

**foreach** **(**DataRow dr **in** ds1**.**Tables**[**"employee"**].**Rows**)**

**{**

EmployeeArtifact temp **=** **new** EmployeeArtifact**();**

temp**.**DateOfBirth **=** dr**[**"dateofbirth"**].**ToString**();**

temp**.**FirstName **=** dr**[**"fname"**].**ToString**();**

temp**.**Address **=** dr**[**"address"**].**ToString**();**

temp**.**Id **=** int**.**Parse**(**dr**[**"id"**].**ToString**());**

temp**.**LastName **=** dr**[**"lname"**].**ToString**();**

temp**.**MiddleName **=** dr**[**"middlename"**].**ToString**();**

temp**.**Designation **=** dr**[**"designation"**].**ToString**();**

temp**.**Qualification **=** dr**[**"qualification"**].**ToString**();**

temp**.**ContactDetails **=** dr**[**"contactdetails"**].**ToString**();**

temp**.**Active **=** bool**.**Parse**(**dr**[**"active"**].**ToString**());**

temp**.**Deleted **=** bool**.**Parse**(**dr**[**"deleted"**].**ToString**());**

empArtifactList**.**Add**(**temp**);**

**}**

**return** empArtifactList**;**

**}**

/// <summary>

/// Updates the employee details for the selected employee in the employee table

/// </summary>

/// <param name="obj">list of the employeedetails</param>

/// <returns>boolean value for whether the update operation is successful or not</returns>

**public** bool EditEmployee**(**EmployeeArtifact obj**)**

**{**

string query **=** "update employee set lname ='" **+** obj**.**LastName **+** "', middlename='" **+** obj**.**MiddleName **+** "' ,designation='" **+** obj**.**Designation

**+** "',address='" **+** obj**.**Address **+** "',contactdetails = '" **+** obj**.**ContactDetails

**+** "',active = '" **+** obj**.**Active **+** "', deleted= '" **+** obj**.**Deleted **+** "', dateofbirth = '" **+** obj**.**DateOfBirth **+** "',qualification = '" **+** obj**.**Qualification **+** "' where id = " **+** obj**.**Id**;**

DbConnectionManager dbConnectionManager **=** **new** DbConnectionManager**();**

SqlConnection connection **=** dbConnectionManager**.**connectToDb**();**

SqlCommand command **=** **new** SqlCommand**(**query**);**

command**.**Connection **=** connection**;**

int result **=** command**.**ExecuteNonQuery**();**

dbConnectionManager**.**diconnectFromDb**(**connection**);**

**if** **(**result **>** 0**)**

**{**

**return** **true;**

**}**

**else**

**{**

**return** **false;**

**}**

**}**

**}**

**}**

### View Employee Form

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class ViewEmployee **:** Form

**{**

**public** ViewEmployee**()**

**{**

InitializeComponent**();**

**}**

**private** void ViewEmployee\_Load**(object** sender**,** EventArgs e**)**

**{**

//retrieves the employee details from the Employee Artifact List and displays in the datagridview

EmployeeArtifact empArtifactObj **=** **new** EmployeeArtifact**();**

List**<**EmployeeArtifact**>** list **=** **new** List**<**EmployeeArtifact**>();**

list **=** empArtifactObj**.**GetAllEmployee**();**

view\_employee**.**DataSource **=** list**;**

**}**

**}**

**}**

### Edit Employee Details Form

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class EditEmployeeDetails **:** Form

**{**

**public** EditEmployeeDetails**()**

**{**

InitializeComponent**();**

**}**

**private** void EditEmployeeDetails\_Load**(object** sender**,** EventArgs e**)**

**{**

//retrieves the list of employee names from the EmployeeArtifact list and shows them in the EmployeeName combobox

EmployeeArtifact empArtifactObj **=** **new** EmployeeArtifact**();**

List**<**EmployeeArtifact**>** list **=** **new** List**<**EmployeeArtifact**>();**

list **=** empArtifactObj**.**GetAllEmployee**();**

empname\_combobox**.**DataSource **=** list**;**

empname\_combobox**.**DisplayMember **=** "FirstName"**;**

empname\_combobox**.**ValueMember **=** "Id"**;**

**}**

/// <summary>

/// retrieves the employee details from the EmployeeArtifact List and displays them in theri respective fieldboxes

/// </summary>

/// <param name="employeename">selected value from the employeename combobox</param>

**private** void DisplayEmployeeDetails**(**string employeename**)**

**{**

EmployeeArtifact selectedEmployee **=** **(**EmployeeArtifact**)**empname\_combobox**.**SelectedItem**;**

empid\_txtbox**.**Text **=** selectedEmployee**.**Id**.**ToString**();**

lname\_txtbox**.**Text **=** selectedEmployee**.**LastName**;**

mname\_txtbox**.**Text **=** selectedEmployee**.**MiddleName**;**

address\_txtbox**.**Text **=** selectedEmployee**.**Address**;**

contact\_txtbox**.**Text **=** selectedEmployee**.**ContactDetails**;**

dob\_pick**.**Value **=** DateTime**.**Parse**(**selectedEmployee**.**DateOfBirth**);**

qualification\_txtbox**.**Text **=** selectedEmployee**.**Qualification**;**

designation\_txtbox**.**Text **=** selectedEmployee**.**Designation**;**

active**.**Checked **=** selectedEmployee**.**Active**;**

leave**.**Checked **=** selectedEmployee**.**Deleted**;**

inactive**.**Checked **=** active**.**Checked **?** **false** **:** **true;**

**}**

**private** void update\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

// on clicking of update button, the values for the respective fields are updated into the EmployeeArtifact List

EmployeeArtifact artifactObj **=** **new** EmployeeArtifact**();**

artifactObj**.**Active **=** active**.**Checked**;**

artifactObj**.**Address **=** address\_txtbox**.**Text**;**

artifactObj**.**ContactDetails **=** contact\_txtbox**.**Text**;**

artifactObj**.**DateOfBirth **=** dob\_pick**.**Value**.**Date**.**ToString**();**

artifactObj**.**Deleted **=** leave**.**Checked **?** **true** **:** **false;**

artifactObj**.**Designation **=** designation\_txtbox**.**Text**;**

artifactObj**.**FirstName **=** empname\_combobox**.**Text**;**

artifactObj**.**MiddleName **=** mname\_txtbox**.**Text**;**

artifactObj**.**LastName **=** lname\_txtbox**.**Text**;**

artifactObj**.**Qualification **=** qualification\_txtbox**.**Text**;**

artifactObj**.**Id **=** int**.**Parse**(**empid\_txtbox**.**Text**);**

**if** **(**artifactObj**.**EditEmployee**(**artifactObj**))**

**{**

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

**}**

**else**

**{**

MessageBox**.**Show**(**"Failed"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void empname\_combobox\_SelectedIndexChanged**(object** sender**,** EventArgs e**)**

**{**

DisplayEmployeeDetails**(**empname\_combobox**.**SelectedValue**.**ToString**());**

**}**

**private** void delete\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

//deletes the selected employee from the employee table in the database

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string query **=** "delete from employee where id = " **+** empid\_txtbox**.**Text**;**

SqlCommand cmd **=** **new** SqlCommand**(**query**);**

cmd**.**Connection **=** conn**;**

int result **=** cmd**.**ExecuteNonQuery**();**

**if** **(**result **>** 0**)**

**{**

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

**}**

**else**

**{**

MessageBox**.**Show**(**"Error"**);**

**}**

**}**

**private** void cancel\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

**this.**Close**();**

**}**

**}**

**}**

### Salary Details Class

This class contains the list to temporarily store the salary details of a particular employee or a group of employees. Further, it also consists of the function to retrieve the salary parameters for the employees, monthly salary for the selected employee for selected date and update the salary parameters.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Data**.**SqlClient**;**

**using** System**.**Data**;**

**using** System**.**ComponentModel**;**

**namespace** Payroll

**{**

**public** class SalaryDetails

**{**

**public** int Id **{** get**;** set**;** **}**

**[**DisplayName**(**"First Name"**)]**

**public** string FirstName **{** get**;** set**;** **}**

**[**DisplayName**(**"Middle Name"**)]**

**public** string MiddleName **{** get**;** set**;** **}**

**[**DisplayName**(**"Last Name"**)]**

**public** string LastName **{** get**;** set**;** **}**

**public** string Designation **{** get**;** set**;** **}**

**[**DisplayName**(**"Basic Salary"**)]**

**public** float BasicSalary **{** get**;** set**;** **}**

**[**DisplayName**(**"Number Of Present Days"**)]**

**public** int NoOfPresentDays **{** get**;** set**;** **}**

**[**DisplayName**(**"Monthly Salary"**)]**

**public** float MonthlySalary **{** get**;** set**;** **}**

**public** float Allowance **{** get**;** set**;** **}**

**public** float Insurance **{** get**;** set**;** **}**

**[**DisplayName**(**"Net Salary"**)]**

**public** float NetSalary **{** get**;** set**;** **}**

**public** List**<**SalaryDetails**>** GetMonthlySalaryDetails**(**string month**,** string year**,** string numberOfDays**)**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string employeeQuery **=** "select \* from employee where active = 'True'"**;**

SqlCommand cmd **=** **new** SqlCommand**(**employeeQuery**);**

cmd**.**Connection **=** conn**;**

SqlDataAdapter da **=** **new** SqlDataAdapter**();**

da**.**SelectCommand **=** cmd**;**

DataSet ds **=** **new** DataSet**();**

da**.**Fill**(**ds**,** "employee"**);**

List**<**SalaryDetails**>** monthlySalaryDetails **=** **new** List**<**SalaryDetails**>();**

**foreach** **(**DataRow dr **in** ds**.**Tables**[**"employee"**].**Rows**)**

**{**

string empid **=** **(**dr**[**"id"**].**ToString**());**

SalaryDetails salary **=** **new** SalaryDetails**();**

AttendanceArtifact selectedMonth **=** **new** AttendanceArtifact**();**

SalaryDetails tempObj **=** salary**.**GetSalaryCredentialsByEmpId**(**empid**);**

salary**.**FirstName **=** dr**[**"fname"**].**ToString**();**

salary**.**MiddleName **=** dr**[**"middlename"**].**ToString**();**

salary**.**LastName **=** dr**[**"lname"**].**ToString**();**

salary**.**Designation **=** dr**[**"designation"**].**ToString**();**

salary**.**Id **=** int**.**Parse**(**empid**);**

salary**.**BasicSalary **=** tempObj**.**BasicSalary**;**

salary**.**NoOfPresentDays **=** int**.**Parse**(**selectedMonth**.**GetAttendanceCountByEmpID**(**empid**,** month**,** year**,** numberOfDays**));**

salary**.**MonthlySalary **=** **(**salary**.**BasicSalary **\*** salary**.**NoOfPresentDays**);**

salary**.**Allowance **=** **(**salary**.**MonthlySalary **\*** **(**tempObj**.**Allowance **/** 100**));**

salary**.**Insurance **=** **(**salary**.**MonthlySalary **\*** **(**tempObj**.**Insurance **/** 100**));**

salary**.**NetSalary **=** **(**salary**.**MonthlySalary **+** salary**.**Allowance **-** salary**.**Insurance**);**

monthlySalaryDetails**.**Add**(**salary**);**

string redundantCheckQuery **=** "select \* from monthlySalary where emp\_id = " **+** int**.**Parse**(**empid**)** **+** " and month = '" **+** year **+** "/" **+** month **+** "/" **+** numberOfDays **+** " 00:00:00.000'"**;**

SqlCommand cmd1 **=** **new** SqlCommand**(**redundantCheckQuery**);**

cmd1**.**Connection **=** conn**;**

SqlDataAdapter da1 **=** **new** SqlDataAdapter**();**

da1**.**SelectCommand **=** cmd1**;**

DataSet ds1 **=** **new** DataSet**();**

da1**.**Fill**(**ds1**,** "monthlySalary"**);**

**if** **(**ds1**.**Tables**[**"monthlySalary"**].**Rows**.**Count **==** 0**)**

**{**

string salaryQuery **=** "insert into monthlySalary(emp\_id,month,monthlySalary,allowance,insurance,netSalary) values( " **+** int**.**Parse**(**empid**)** **+** ",'" **+** year **+** "/" **+** month **+** "/" **+** numberOfDays **+** " 00:00:00.000' ," **+** salary**.**MonthlySalary **+** "," **+** salary**.**Allowance **+** "," **+** salary**.**Insurance **+** "," **+** salary**.**NetSalary **+** ")"**;**

SqlCommand command **=** **new** SqlCommand**(**salaryQuery**);**

command**.**Connection **=** conn**;**

command**.**ExecuteNonQuery**();**

**}**

**}**

connect**.**diconnectFromDb**(**conn**);**

**return** monthlySalaryDetails**;**

**}**

**public** SalaryDetails GetMonthlySalaryByEmpId**(**string empid**,** string month**,** string year**,** string numberOfDays**)**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string salaryQuery **=** "select \* from monthlySalary where emp\_id = " **+** empid **+** " and month = '" **+** year **+** "/" **+** month **+** "/" **+** numberOfDays **+** " 00:00:00.000'"**;**

SqlCommand cmd **=** **new** SqlCommand**(**salaryQuery**);**

cmd**.**Connection **=** conn**;**

SqlDataAdapter da **=** **new** SqlDataAdapter**();**

da**.**SelectCommand **=** cmd**;**

DataSet ds **=** **new** DataSet**();**

da**.**Fill**(**ds**,** "monthlySalary"**);**

List**<**SalaryDetails**>** salaryArtifactList **=** **new** List**<**SalaryDetails**>();**

SalaryDetails salary **=** **new** SalaryDetails**();**

AttendanceArtifact selectedMonth **=** **new** AttendanceArtifact**();**

**foreach** **(**DataRow dr **in** ds**.**Tables**[**"monthlySalary"**].**Rows**)**

**{**

salary**.**Id **=** int**.**Parse**(**empid**);**

salary**.**MonthlySalary **=** float**.**Parse**(**dr**[**"monthlySalary"**].**ToString**());**

salary**.**Allowance **=** float**.**Parse**(**dr**[**"allowance"**].**ToString**());**

salary**.**Insurance **=** float**.**Parse**(**dr**[**"insurance"**].**ToString**());**

salary**.**NetSalary **=** float**.**Parse**(**dr**[**"netSalary"**].**ToString**());**

salaryArtifactList**.**Add**(**salary**);**

**}**

**return** salary**;**

**}**

**public** SalaryDetails GetSalaryCredentialsByEmpId**(**string empId**)**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string salaryQuery **=** "select \* from salary where employee\_id = " **+** empId**;**

SqlCommand cmd **=** **new** SqlCommand**(**salaryQuery**);**

cmd**.**Connection **=** conn**;**

SqlDataAdapter da **=** **new** SqlDataAdapter**();**

da**.**SelectCommand **=** cmd**;**

DataSet ds **=** **new** DataSet**();**

da**.**Fill**(**ds**,** "salary"**);**

List**<**SalaryDetails**>** salaryArtifactList **=** **new** List**<**SalaryDetails**>();**

SalaryDetails salary **=** **new** SalaryDetails**();**

**foreach** **(**DataRow dr **in** ds**.**Tables**[**"salary"**].**Rows**)**

**{**

salary**.**Id **=** int**.**Parse**((**dr**[**"employee\_id"**].**ToString**()));**

salary**.**BasicSalary **=** float**.**Parse**(**dr**[**"basic\_salary"**].**ToString**());**

salary**.**Allowance **=** float**.**Parse**(**dr**[**"allowance"**].**ToString**());**

salary**.**Insurance **=** float**.**Parse**(**dr**[**"insurance"**].**ToString**());**

salaryArtifactList**.**Add**(**salary**);**

**}**

connect**.**diconnectFromDb**(**conn**);**

**return** salary**;**

**}**

**public** bool UpdateSalaryDetails**(**SalaryDetails obj**)**

**{**

DbConnectionManager dbConnectionManager **=** **new** DbConnectionManager**();**

SqlConnection connection **=** dbConnectionManager**.**connectToDb**();**

string query **=** "update salary set basic\_salary =" **+** obj**.**BasicSalary **+** " ,allowance =" **+** obj**.**Allowance **+** " ,insurance =" **+** obj**.**Insurance **+** " where employee\_id =" **+** obj**.**Id**;**

SqlCommand command **=** **new** SqlCommand**(**query**);**

command**.**Connection **=** connection**;**

int result **=** command**.**ExecuteNonQuery**();**

**if** **(**result **>** 0**)**

**{**

**return** **true;**

**}**

**else**

**{**

**return** **false;**

**}**

**}**

**}**

**}**

### Update Salary Parameters Form

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Data**;**

**using** System**.**Drawing**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class SalaryCredentialAssigner **:** Form

**{**

**public** SalaryCredentialAssigner**()**

**{**

InitializeComponent**();**

**}**

**private** void SalaryCredentialAssigner\_Load**(object** sender**,** EventArgs e**)**

**{**

// retrieves the employee names from the EmployeeArtifact List and displays them in the employeename combobox.

EmployeeArtifact empArtifactObj **=** **new** EmployeeArtifact**();**

List**<**EmployeeArtifact**>** list **=** **new** List**<**EmployeeArtifact**>();**

list **=** empArtifactObj**.**GetAllEmployee**();**

empname\_combobox**.**DataSource **=** list**;**

empname\_combobox**.**DisplayMember **=** "FirstName"**;**

empname\_combobox**.**ValueMember **=** "Id"**;**

**}**

**private** void DisplayEmployeeDetails**(object** p**)**

**{**

**throw** **new** NotImplementedException**();**

**}**

/// <summary>

/// retrieves and displays the employee details and their respective salary from the EmployeeArtifact and SalaryDetails Lists using the employeename

/// </summary>

/// <param name="employeename">selected value from the employeename combobox </param>

**private** void DisplayEmployeeDetails**(**string employeename**)**

**{**

EmployeeArtifact selectedEmployee **=** **(**EmployeeArtifact**)**empname\_combobox**.**SelectedItem**;**

SalaryDetails salaryParameter **=** **new** SalaryDetails**();**

SalaryDetails tempObj **=** salaryParameter**.**GetSalaryCredentialsByEmpId**(**selectedEmployee**.**Id**.**ToString**());**

emp\_idtxtbox**.**Text **=** selectedEmployee**.**Id**.**ToString**();**

emplname\_txtbox**.**Text **=** selectedEmployee**.**LastName**;**

designation\_txtbox**.**Text **=** selectedEmployee**.**Designation**;**

basicSalary\_txt**.**Text **=** tempObj**.**BasicSalary**.**ToString**();**

allowance\_txt**.**Text **=** tempObj**.**Allowance**.**ToString**();**

insurance\_txt**.**Text **=** tempObj**.**Insurance**.**ToString**();**

**}**

**private** void empid\_combobox\_SelectedIndexChanged**(object** sender**,** EventArgs e**)**

**{**

DisplayEmployeeDetails**(**empname\_combobox**.**Text**);**

**}**

**private** void Update\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

//Updates the new values to the SalaryDetails list from their respective fieldboxes and displays messages accordingly

SalaryDetails artifactObj **=** **new** SalaryDetails**();**

artifactObj**.**Id **=** int**.**Parse**(**emp\_idtxtbox**.**Text**);**

artifactObj**.**BasicSalary **=** float**.**Parse**(**basicSalary\_txt**.**Text**);**

artifactObj**.**Insurance **=** float**.**Parse**(**insurance\_txt**.**Text**);**

artifactObj**.**Allowance **=** float**.**Parse**(**allowance\_txt**.**Text**);**

**if** **(**float**.**Parse**(**allowance\_txt**.**Text**)** **>** 100 **||** float**.**Parse**(**insurance\_txt**.**Text**)** **>** 100**)**

**{**

MessageBox**.**Show**(**"Invalid values entered"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**else**

**{**

**if** **(**artifactObj**.**UpdateSalaryDetails**(**artifactObj**))**

**{**

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

**}**

**else**

**{**

MessageBox**.**Show**(**"Failed"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**}**

**private** void basicSalary\_txt\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

const char delete **=** **(**char**)**0x08**;**

**if** **(!**char**.**IsNumber**(**e**.**KeyChar**)** **&&** e**.**KeyChar **!=** delete**)**

**{**

e**.**Handled **=** **true;**

MessageBox**.**Show**(**"The value for this field must be a number"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**}**

**private** void allowance\_txt\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

const char delete **=** **(**char**)**0x08**;**

**if** **(!**char**.**IsNumber**(**e**.**KeyChar**)** **&&** e**.**KeyChar **!=** delete**)**

**{**

e**.**Handled **=** **true;**

MessageBox**.**Show**(**"The value for this field must be a number"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**else** **if** **(**int**.**Parse**(**allowance\_txt**.**Text**)** **>** 100**)**

**{**

MessageBox**.**Show**(**"The value for this field must be less than 100"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**}**

**private** void insurance\_txt\_KeyPress**(object** sender**,** KeyPressEventArgs e**)**

**{**

const char delete **=** **(**char**)**0x08**;**

**if** **(!**char**.**IsNumber**(**e**.**KeyChar**)** **&&** e**.**KeyChar **!=** delete**)**

**{**

e**.**Handled **=** **true;**

MessageBox**.**Show**(**"The value for this field must be a number"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**else** **if** **(**int**.**Parse**(**insurance\_txt**.**Text**)** **>** 100**)**

**{**

MessageBox**.**Show**(**"The value for this field must be less than 100"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**}**

**}**

**}**

### Attendance Artifact Class

This class contains the list to temporarily store the attendance details of a particular employee or a group of employees. It contains functions that return the number of attended days a selected employee or a group of employees.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Data**;**

**using** System**.**Data**.**SqlClient**;**

**using** System**.**ComponentModel**;**

**namespace** Payroll

**{**

class AttendanceArtifact

**{**

**[**DisplayName**(**"Staff Id"**)]**

**public** int Id **{** get**;** set**;** **}**

**[**DisplayName**(**"First Name"**)]**

**public** string FirstName **{** get**;** set**;** **}**

**[**DisplayName**(**"Middle Name"**)]**

**public** string MiddleName **{** get**;** set**;** **}**

**[**DisplayName**(**"Last Name"**)]**

**public** string LastName **{** get**;** set**;** **}**

**[**DisplayName**(**"Contact Number"**)]**

**public** string ContactDetails **{** get**;** set**;** **}**

**public** string Designation **{** get**;** set**;** **}**

**[**DisplayName**(**"Number Of Present Days"**)]**

**public** string NumberOfPresentDays **{** get**;** set**;** **}**

/// <summary>

/// Retrieves the number of present days for a selected employee for the selected date

/// </summary>

/// <param name="empid">the employee id of the selected employee</param>

/// <param name="month">the value for the selected month</param>

/// <param name="year">The value for the selected year</param>

/// <param name="numberOfDays">the value for the number of days in the selected month</param>

/// <returns>the number of days the selected employee was present in the month</returns>

**public** string GetAttendanceCountByEmpID**(**string empid**,** string month**,** string year**,** string numberOfDays**)**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string attendanceQuery **=** "select \* from attendance where employee\_id = " **+** empid **+** " and attendance\_date between '" **+** year **+** "/" **+** month **+** "/1 00:00:00.000' and '" **+** year **+** "/" **+** month **+** "/" **+** numberOfDays **+** " 00:00:00.000' and present = 'True' "**;**

SqlCommand cmd1 **=** **new** SqlCommand**(**attendanceQuery**);**

cmd1**.**Connection **=** conn**;**

SqlDataAdapter da1 **=** **new** SqlDataAdapter**();**

da1**.**SelectCommand **=** cmd1**;**

DataSet ds1 **=** **new** DataSet**();**

da1**.**Fill**(**ds1**,** "attendance"**);**

**return** ds1**.**Tables**[**"attendance"**].**Rows**.**Count**.**ToString**();**

**}**

/// <summary>

/// Retrieves all the active employees from the employee table and stores them in the AttendanceArtifact List

/// along with their number of present days for the selected date

/// </summary>

/// <param name="month">the value for the selected month</param>

/// <param name="year">the value for the selected year</param>

/// <param name="numberOfDays">number of days in the selected month</param>

/// <returns></returns>

**public** List**<**AttendanceArtifact**>** GetAttendanceCount**(**string month**,** string year**,** string numberOfDays**)**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string noOfDays **=** numberOfDays**;**

string employeeQuery **=** "select \* from employee where active = 'True' "**;**

SqlCommand cmd **=** **new** SqlCommand**(**employeeQuery**);**

cmd**.**Connection **=** conn**;**

SqlDataAdapter da **=** **new** SqlDataAdapter**();**

da**.**SelectCommand **=** cmd**;**

DataSet ds **=** **new** DataSet**();**

da**.**Fill**(**ds**,** "employee"**);**

List**<**AttendanceArtifact**>** attendanceArtifactList **=** **new** List**<**AttendanceArtifact**>();**

**foreach** **(**DataRow dr **in** ds**.**Tables**[**"employee"**].**Rows**)**

**{**

string empid **=** **(**dr**[**"id"**].**ToString**());**

string attendanceQuery **=** "select \* from attendance where employee\_id = " **+** empid **+** " and attendance\_date between '" **+** year **+** "/" **+** month **+** "/1 00:00:00.000' and '" **+** year **+** "/" **+** month **+** "/" **+** noOfDays **+** " 00:00:00.000' and present = 'True'"**;**

SqlCommand cmd1 **=** **new** SqlCommand**(**attendanceQuery**);**

cmd1**.**Connection **=** conn**;**

SqlDataAdapter da1 **=** **new** SqlDataAdapter**();**

da1**.**SelectCommand **=** cmd1**;**

DataSet ds1 **=** **new** DataSet**();**

da1**.**Fill**(**ds1**,** "attendance"**);**

AttendanceArtifact attendance **=** **new** AttendanceArtifact**();**

attendance**.**Id **=** int**.**Parse**(**empid**);**

attendance**.**FirstName **=** dr**[**"fname"**].**ToString**();**

attendance**.**MiddleName **=** dr**[**"middlename"**].**ToString**();**

attendance**.**LastName **=** dr**[**"lname"**].**ToString**();**

attendance**.**Designation **=** dr**[**"designation"**].**ToString**();**

attendance**.**ContactDetails **=** dr**[**"contactdetails"**].**ToString**();**

attendance**.**NumberOfPresentDays **=** ds1**.**Tables**[**"attendance"**].**Rows**.**Count**.**ToString**();**

attendanceArtifactList**.**Add**(**attendance**);**

connect**.**diconnectFromDb**(**conn**);**

**}**

**return** attendanceArtifactList**;**

**}**

**}**

**}**

### Month Class

This class comprises of a list of months with their IDs and the number of days in each month.

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**namespace** Payroll

**{**

**public** class Month

**{**

**public** string Id **{** get**;** set**;** **}**

**public** string Name **{** get**;** set**;** **}**

**public** string NumberOfDays **{** get**;** set**;** **}**

/// <summary>

/// Stores the list of the month, their id and number of days in it

/// </summary>

/// <returns>the list of month with its id and number of days</returns>

**public** List**<**Month**>** GetMonths**()**

**{**

List**<**Month**>** listOfMonth **=** **new** List**<**Month**>();**

Month obj **=** **new** Month**();**

obj**.**Id **=** "1"**;**

obj**.**Name **=** "January"**;**

obj**.**NumberOfDays **=** "31"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "2"**;**

obj**.**Name **=** "February"**;**

obj**.**NumberOfDays **=** "28"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "3"**;**

obj**.**Name **=** "March"**;**

obj**.**NumberOfDays **=** "31"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "4"**;**

obj**.**Name **=** "April"**;**

obj**.**NumberOfDays **=** "30"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "5"**;**

obj**.**Name **=** "May"**;**

obj**.**NumberOfDays **=** "31"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "6"**;**

obj**.**Name **=** "June"**;**

obj**.**NumberOfDays **=** "30"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "7"**;**

obj**.**Name **=** "July"**;**

obj**.**NumberOfDays **=** "31"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "8"**;**

obj**.**Name **=** "August"**;**

obj**.**NumberOfDays **=** "30"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "9"**;**

obj**.**Name **=** "September"**;**

obj**.**NumberOfDays **=** "31"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "10"**;**

obj**.**Name **=** "October"**;**

obj**.**NumberOfDays **=** "30"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "11"**;**

obj**.**Name **=** "November"**;**

obj**.**NumberOfDays **=** "31"**;**

listOfMonth**.**Add**(**obj**);**

obj **=** **new** Month**();**

obj**.**Id **=** "12"**;**

obj**.**Name **=** "December"**;**

obj**.**NumberOfDays **=** "30"**;**

listOfMonth**.**Add**(**obj**);**

**return** listOfMonth**;**

**}**

**}**

**}**

### Daily Attendance User Control

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Drawing**;**

**using** System**.**Data**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class DailyAttendance **:** UserControl

**{**

**public** DailyAttendance**()**

**{**

InitializeComponent**();**

**}**

**private** void comboBox1\_SelectedIndexChanged**(object** sender**,** EventArgs e**)**

**{**

Notice**.**Text **=** ""**;**

DisplayEmployeeDetails**(**empfname\_combobox**.**Text**);**

**}**

**private** void DailyAttendance\_Load**(object** sender**,** EventArgs e**)**

**{**

//Displays the list of active employees in the employeename combobox from the EmployeeArtifact List

EmployeeArtifact empArtifactObj **=** **new** EmployeeArtifact**();**

List**<**EmployeeArtifact**>** list **=** **new** List**<**EmployeeArtifact**>();**

list **=** empArtifactObj**.**GetActiveEmployee**();**

empfname\_combobox**.**DataSource **=** list**;**

empfname\_combobox**.**DisplayMember **=** "FirstName"**;**

empfname\_combobox**.**ValueMember **=** "Id"**;**

**}**

/// <summary>

/// Displays the employeeid, lastname and designation of the selected employee in their respected fields

/// by retrieving them from the EmployeeArtifact List

/// </summary>

/// <param name="employeename">selected value from the employee name combobox</param>

**private** void DisplayEmployeeDetails**(**string employeename**)**

**{**

EmployeeArtifact selectedEmployee **=** **(**EmployeeArtifact**)**empfname\_combobox**.**SelectedItem**;**

empid\_txtbox**.**Text **=** selectedEmployee**.**Id**.**ToString**();**

emplname\_txtbox**.**Text **=** selectedEmployee**.**LastName**;**

designation\_txtbox**.**Text **=** selectedEmployee**.**Designation**;**

**}**

/// <summary>

/// checks if the data to be added already exists in the table or not and if not adds the corresponding values for the selected employee

/// </summary>

/// <param name="empid">employee id of the selected employee</param>

**private** void AddAttendanceDetails**(**string empid**)**

**{**

DbConnectionManager connect **=** **new** DbConnectionManager**();**

SqlConnection conn **=** connect**.**connectToDb**();**

string query1 **=** "select \* from attendance where employee\_id = " **+** empid **+** " and attendance\_date = '" **+** dateTimePicker1**.**Value**.**Date **+** "'"**;**

SqlCommand cmd1 **=** **new** SqlCommand**(**query1**);**

cmd1**.**Connection **=** conn**;**

SqlDataAdapter da1 **=** **new** SqlDataAdapter**();**

da1**.**SelectCommand **=** cmd1**;**

DataSet ds1 **=** **new** DataSet**();**

da1**.**Fill**(**ds1**,** "attendance"**);**

**if** **(**ds1**.**Tables**[**"attendance"**].**Rows**.**Count **!=** 0**)**

**{**

Notice**.**Text **=** "Attendance Already Marked"**;**

**}**

**else**

**{**

**if** **(**attendanceMarked**(**empid**,** conn**)** **==** **true)**

**{**

Notice**.**Text **=** "Attendance Marked Successfully"**;**

**}**

**else**

**{**

Notice**.**Text **=** "Error! Try Again!"**;**

**}**

**}**

connect**.**diconnectFromDb**(**conn**);**

**}**

/// <summary>

/// inserts the attendance records for the selected employee to the attendance table

/// </summary>

/// <param name="empid">employee id of the selected employee</param>

/// <param name="conn">SQL connection string for the ongoing connection</param>

/// <returns>boolean value for success or failure of operation</returns>

**public** bool attendanceMarked**(**string empid**,** SqlConnection conn**)**

**{**

//assigns a boolean value for the selected value

bool isPresent**;**

isPresent **=** **false;**

**if** **(**attendance\_combobox**.**SelectedIndex **==** **-**1**)**

**{**

MessageBox**.**Show**(**"Please select an option!"**,** "Validation Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Warning**);**

**}**

**else**

**{**

**if** **(**attendance\_combobox**.**SelectedItem**.**ToString**()** **==** "Present"**)**

**{**

isPresent **=** **true;**

**}**

**else** **if** **(**attendance\_combobox**.**SelectedItem**.**ToString**()** **==** "Absent"**)**

**{**

isPresent **=** **false;**

**}**

string query **=** "insert into attendance(employee\_id,attendance\_date,present) values(" **+** empid **+** ",'" **+** dateTimePicker1**.**Value**.**Date **+** "','" **+** isPresent **+** "')"**;**

SqlCommand command **=** **new** SqlCommand**(**query**);**

command**.**Connection **=** conn**;**

int result **=** command**.**ExecuteNonQuery**();**

**if** **(**result **>** 0**)**

**{**

**return** **true;**

**}**

**else**

**{**

**return** **false;**

**}**

**}**

**return** **false;**

**}**

**private** void done\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

//checks if the selected date exceeds today's date and displays error in the negtive case

**if** **(**dateTimePicker1**.**Value**.**Date **<=** System**.**DateTime**.**Today**.**Date**)**

**{**

AddAttendanceDetails**(**empid\_txtbox**.**Text**);**

**}**

**else**

**{**

MessageBox**.**Show**(**"The selected date exceeds today's date"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**}**

**}**

### Monthly Attendance Report User Control

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Drawing**;**

**using** System**.**Data**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class MonthlyAttendanceReport **:** UserControl

**{**

**public** MonthlyAttendanceReport**()**

**{**

InitializeComponent**();**

**}**

**private** void MonthlyAttendanceReport\_Load**(object** sender**,** EventArgs e**)**

**{**

selectYear\_combobox**.**SelectedIndex **=** 0**;**

//Displays the list of months in the combobox by retrieveing the values from the Month List

Month monthObj **=** **new** Month**();**

selectMonth\_combobox**.**DataSource **=** monthObj**.**GetMonths**();**

selectMonth\_combobox**.**DisplayMember **=** "Name"**;**

selectMonth\_combobox**.**ValueMember **=** "Id"**;**

**}**

**private** void show\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

//retrieves the list of attendance details about all emplyees for the selected month and displays them in the datagrid view

AttendanceArtifact attendance **=** **new** AttendanceArtifact**();**

Month selectedmonth **=** **(**Month**)**selectMonth\_combobox**.**SelectedItem**;**

List**<**AttendanceArtifact**>** list **=** **new** List**<**AttendanceArtifact**>();**

list **=** attendance**.**GetAttendanceCount**(**selectedmonth**.**Id**.**ToString**(),** selectYear\_combobox**.**SelectedItem**.**ToString**(),** selectedmonth**.**NumberOfDays**.**ToString**());**

dataGridView1**.**DataSource **=** list**;**

**}**

**}**

**}**

### Monthly Salary Sheet User Control

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Drawing**;**

**using** System**.**Data**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**namespace** Payroll

**{**

**public** partial class MonthlySalarySheet **:** UserControl

**{**

**public** MonthlySalarySheet**()**

**{**

InitializeComponent**();**

**}**

**private** void show\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

//Retrieves the list of salary for all employees for the selected date and displays them in the datagrid view

SalaryDetails salary **=** **new** SalaryDetails**();**

Month selectedmonth **=** **(**Month**)**selectMonth\_combobox**.**SelectedItem**;**

List**<**SalaryDetails**>** list **=** **new** List**<**SalaryDetails**>();**

**if** **(**DateTime**.**Parse**(**selectedmonth**.**Id **+** "/" **+** selectedmonth**.**NumberOfDays **+** "/" **+** selectYear\_combobox**.**SelectedItem**.**ToString**())** **<=** System**.**DateTime**.**Today**.**Date**)**

**{**

list **=** salary**.**GetMonthlySalaryDetails**(**selectedmonth**.**Id**.**ToString**(),** selectYear\_combobox**.**SelectedItem**.**ToString**(),** selectedmonth**.**NumberOfDays**.**ToString**());**

dataGridView1**.**DataSource **=** list**;**

**}**

**else**

**{**

MessageBox**.**Show**(**"The selected date exceeds today's date"**,** "Error"**,** MessageBoxButtons**.**OK**,** MessageBoxIcon**.**Error**);**

**}**

**}**

**private** void MonthlySalarySheet\_Load**(object** sender**,** EventArgs e**)**

**{**

selectYear\_combobox**.**SelectedIndex **=** 0**;**

//Retrieves the list of the month form the Month List and displays them in the selectmonth cpmbobox

Month monthObj **=** **new** Month**();**

selectMonth\_combobox**.**DataSource **=** monthObj**.**GetMonths**();**

selectMonth\_combobox**.**DisplayMember **=** "Name"**;**

selectMonth\_combobox**.**ValueMember **=** "Id"**;**

**}**

**}**

**}**

### Perspective Employee PaySlips User Control

**using** System**;**

**using** System**.**Collections**.**Generic**;**

**using** System**.**ComponentModel**;**

**using** System**.**Drawing**;**

**using** System**.**Data**;**

**using** System**.**Linq**;**

**using** System**.**Text**;**

**using** System**.**Windows**.**Forms**;**

**using** System**.**Data**.**SqlClient**;**

**namespace** Payroll

**{**

**public** partial class PerspectivePayslip **:** UserControl

**{**

**public** PerspectivePayslip**()**

**{**

InitializeComponent**();**

**}**

**private** void PerspectivePayslip\_Load**(object** sender**,** EventArgs e**)**

**{**

//Retrieves the list of month and active employees and displays them in their respected comboboxes from the Month and EmployeeArtifact List

selectYear\_combobox**.**SelectedIndex **=** 0**;**

Month monthObj **=** **new** Month**();**

selectMonth\_combobox**.**DataSource **=** monthObj**.**GetMonths**();**

selectMonth\_combobox**.**DisplayMember **=** "Name"**;**

selectMonth\_combobox**.**ValueMember **=** "Id"**;**

EmployeeArtifact empArtifactObj **=** **new** EmployeeArtifact**();**

List**<**EmployeeArtifact**>** list **=** **new** List**<**EmployeeArtifact**>();**

list **=** empArtifactObj**.**GetActiveEmployee**();**

fname\_combobox**.**DataSource **=** list**;**

fname\_combobox**.**DisplayMember **=** "FirstName"**;**

fname\_combobox**.**ValueMember **=** "Id"**;**

**}**

/// <summary>

/// Retrievs the monthlySalary details of the selected employee for the slected date and displays them in their respective fieldboxes

/// </summary>

**private** void DisplayEmployeeDetails**()**

**{**

EmployeeArtifact selectedEmployee **=** **(**EmployeeArtifact**)**fname\_combobox**.**SelectedItem**;**

lname\_txtbox**.**Text **=** selectedEmployee**.**LastName**;**

Month selectedMonth **=** **(**Month**)**selectMonth\_combobox**.**SelectedItem**;**

SalaryDetails salaryParameter **=** **new** SalaryDetails**();**

SalaryDetails tempObj **=** salaryParameter**.**GetMonthlySalaryByEmpId**(**selectedEmployee**.**Id**.**ToString**(),** selectedMonth**.**Id**.**ToString**(),** selectYear\_combobox**.**SelectedItem**.**ToString**(),** selectedMonth**.**NumberOfDays**.**ToString**());**

emp\_idtxtbox**.**Text **=** selectedEmployee**.**Id**.**ToString**();**

designation\_txtbox**.**Text **=** selectedEmployee**.**Designation**;**

monthlySalary\_txtbox**.**Text **=** tempObj**.**MonthlySalary**.**ToString**();**

netSalary\_txtbox**.**Text **=** tempObj**.**NetSalary**.**ToString**();**

allowance\_txt**.**Text **=** tempObj**.**Allowance**.**ToString**();**

insurance\_txt**.**Text **=** tempObj**.**Insurance**.**ToString**();**

**}**

**private** void show\_button\_Click**(object** sender**,** EventArgs e**)**

**{**

DisplayEmployeeDetails**();**

**}**

**}**

**}**

# 4.1.9 System Maintenance

As the system is tailored software, there might be some adaptation needed to be considered as time passes. Further, some tweaks must be made in future to avoid obsolescence of the system.

Adaptive maintenance will be done when conditions change from those that existed when the original system was created. The calendar for the attendance is limited for a few years. This feature must be updated. There are various other features that might need consideration for adaptive maintenance

Perfective maintenance will be done when new feature for the better efficiency of the systems is to be added. The feature of the system might get slow. So, perfective maintenance is made in this purpose to increase the speed of the system. Search operations, and designs need perfective maintenance.