**Cambridge International Examinations Computing 9691 May/June 2013**

2013

**Payroll Project**

**Capital Marketing System Pvt. Ltd.**

**Bishal Regmi Candidate Number: Centre: Lumbini International College (NP716)**

Table of Contents

[1.1. Introduction and Nature of the Problem 4](#_Toc353017639)

[1.1.1 Brief description of the organization 4](#_Toc353017640)

[1.1.2 The Client and the User 4](#_Toc353017641)

[1.1.3 Statement of the problem: 4](#_Toc353017642)

[1.1.4 Current methods used: 4](#_Toc353017643)

[1.1.5 Origin and Forms of Data: 5](#_Toc353017644)

[1.2. Investigation and Analysis 6](#_Toc353017647)

[1.2.1 Investigation Plan: 6](#_Toc353017648)

[1.2.2 Interview 1: 6](#_Toc353017649)

[1.2.3 Observations from the current system: 8](#_Toc353017650)

[1.2.4 Description of the Existing System: 9](#_Toc353017653)

[1.2.5 Interview 2: 11](#_Toc353017654)

[1.2.6 Problems in the Existing System: 12](#_Toc353017655)

[1.2.7 Alternative Solutions: 12](#_Toc353017656)

[1.2.8 Proposed Solution: 14](#_Toc353017657)

[1.2.9 Requirements Specification: 15](#_Toc353017658)

[1.2.9.1 Functional Requirements 15](#_Toc353017659)

[1.2.9.2 Input Requirements 15](#_Toc353017660)

[1.2.9.3 Output Requirements 15](#_Toc353017661)

[1.2.9.4 Software Requirements 15](#_Toc353017662)

[1.2.9.5 Hardware Requirements 16](#_Toc353017663)

[2.1 Objectives 17](#_Toc353017664)

[2.2 Nature of the Solution 18](#_Toc353017665)

[2.2.1 Proposed File Record and Data Structure 18](#_Toc353017666)

[2.2.2 Detailed Systems Design 18](#_Toc353017667)

[2.2.3 Tables 19](#_Toc353017668)

[2.2.4 Input Design 22](#_Toc353017674)

[2.2.5 Output Design 27](#_Toc353017681)

[2.2.6 Menu Design 29](#_Toc353017686)

[2.2.7 System Flowchart 31](#_Toc353017687)

[2.2.8 Data Flow Diagrams 31](#_Toc353017688)

[2.3 Intended Benefits 32](#_Toc353017689)

[2.4. Limits of the Scope of the Solution 32](#_Toc353017690)

[2.4.1 System Limitations 32](#_Toc353017691)

[2.4.2 File Size Estimation 33](#_Toc353017692)

[3.1 Software Development 34](#_Toc353017693)

[3.1.1 Code Listing 34](#_Toc353017694)

[3.1.2 Development of Database 72](#_Toc353017714)

[3.1.3 Development of Input Methods 76](#_Toc353017720)

[3.1.4 Development of Output Methods 82](#_Toc353017727)

[3.1.5 Development of Database Queries/Filters 86](#_Toc353017732)

[3.2 Testing 93](#_Toc353017744)

[3.2.1 Testing Strategy 93](#_Toc353017745)

[3.2.2 Test Plan 94](#_Toc353017746)

[3.2.3 Testing 97](#_Toc353017747)

[3.3 Implementation 109](#_Toc353017776)

[3.3.1 Implementation Plan 109](#_Toc353017777)

[3.3.2 User Training 109](#_Toc353017778)

[3.3.3 User Testing 109](#_Toc353017779)

[3.3.4 User Acceptance 110](#_Toc353017780)

[3.4. Appropriateness of Structure and Exploitation of Available Resources 111](#_Toc353017781)

[3.4.1. Use of Available Resources 111](#_Toc353017782)

[3.4.2. Problem Log 111](#_Toc353017785)

[5.1 Degree of Success 112](#_Toc353017786)

[5.2 Evaluation of User’s Response 113](#_Toc353017787)

[5.3 Desirable Extensions 113](#_Toc353017788)

1. **Definition, Investigation And Analysis**

# Introduction and Nature of the Problem

## 1.1.1 Brief description of the organization

Capital Market Solution Nepal Pvt. Ltd. (CAPMARK Nepal) is a brokerage firm with financial advisory service. It approaches traders to deal with online trading on international commodities through registered exchanges of Nepal. They have 3 separate departments viz. Trading (24 hour online), Research & Development (R&D) and Customer Support Desk (CSD) for their valuable investors to get succeed in commodity market trading.

## 1.1.2 The Client and the User

Mr. Paudel is Mr. Sudersun Paudel, the Executive officer of Capital Marketing Systems situated at MinBhawan, Kathmandu, Nepal. The user of the program is the receptionist. He/she keeps the records and then processes the further calculations.

## 1.1.3 Statement of the problem:

Mr. Paudel has a problem in keeping the record of the attendance of his staffs. Due to this, he has a problem in calculating the monthly salary of his officers. As the salary calculation is based on the days present in the office, the company has been in a problem to calculate the monthly salary of the peers working in the office as it is has been a tedious process at present. Therefore, I am here to abet the organization by designing a program that records the attendance and followed by payroll system for the salary calculation.

## 1.1.4 Current methods used:

At present, the staffs of the organization mark their presence by signing on the register placed at the reception. At the time of calculating the monthly salary, the accountant has to turn up the pile of register and count the days the staffs are present and calculate accordingly through Microsoft Excel. The accountant then rechecks the calculations to avoid errors in calculation.

## 1.1.5 Origin and Forms of Data:

The data is collected in two processes. First the total no of attendance from the register is calculated. Then it is sent to the accounting system where the further payroll calculations are done.

### Attendance Register:

The register contains following fields:

1. Staff Id – A unique Id no which represent an staff
2. Staff Name – The name of the staff whose details are mentioned
3. Date – The specific date at which the attendance was taken
4. Designation – The designation of the staff at the office.
5. Signature – The indicator that marks the concerned staff as present
6. Remarks – The field where the reason for being absent or other parameters are specified.

### The Salary Calculation Sheet:

The calculations of the salary is presently done in Microsoft Excel which is saved for future use and also printed and provided to the respective staff along with their salary.

The Calculation Sheet contains the following fields:

1. Date – The specific date when the salary was calculated
2. Salary Slip – The month for which the salary was calculated
3. Basic Pay – The basic salary of the staff
4. Allowance – The allowance provided to the staff for that month
5. Bonus – The company share bonus for the respective staff
6. Tax – The amount of tax for that month
7. No of days present – The no of days the staff had reported the office that month
8. Deduction – The amount of salary deducted as penalty for absent days
9. Net Salary – The net salary to be given after all calculations have been made

In addition to these, the company has an archived file of all it staffs who no longer work in the office. It contains their basic information and contact details.

All the data mentioned above are collected and input by the receptionist or an accountant. He/she makes daily mark of attendance of the staff present at the office and finally the data is input into the calculation sheet where the further calculation is made by applying formulae at Microsoft Excel.

# 1.2. Investigation and Analysis

## 1.2.1 Investigation Plan:

At present, every activity in the office is done manually. The attendance is non-computerized. Although the problem is defined easily, a comprehensive study is required in order to find that how the system works and what are the difficulties faced by the customers. For that, interviews with Mr. Paudel will be the first and foremost approach. I will conduct various interview sessions with Mr. Paudel to crack the overall problems of the system and their solutions accordingly. In addition to this, I will also make observations on the current system during working hours so that I will be able to make a clear and thorough investigation. I will also make some minor interactions with the staffs in the office to collect information, suggestions and recommendations for the new system. Finally, I will code the new system accordingly; then I will conclude the problems and their solutions and ask Mr. Paudel to choose the best way out according to his limitations.

## 1.2.2 Interview 1:

Date : 8th January 2013

Q: What are the problems in the current way your system is being dealt with?

A: *The current process has been a really tedious task. First of all we have no computerized system due to which I am interested in a computerized approach. Whenever a staff enters, he has to provide his signature on the attendance register. Then at the time of calculation of monthly statements, the accountant has to count all the present marks and then calculate accordingly. I want a software that does all these task as a whole without any arduous activities to be done.*

Q: What sort of functions would you like to have in your new system?

A: *The system should accomplish and solve all my tasks and problems I have specified. It should be fast and efficient. Make sure you add the calculation and notes making and other essential add-ons on the system so that the operator need not to indulge in any other software else than this system at the time of his/her work.*

## 1.2.3 Observations from the current system:

In order to get a clear image of the system and its existing problems, I made a visit to the office on one of its working days and also on the last day of the month when the salary is calculated. This aided me in identifying the problem more clearly and provided me further information and idea to amend in the new system.

The observations I made are as follows:

### Attendance Process:

* A staff enters into the office.
* He makes his signature and entry time on the attendance register placed on the table next to the door.
* When he is about to get out from the office, he marks his exit time on the register and leaves.
* The same process implies for all the staffs.

### The Payroll Process:

* The perspective attendance records of all the staffs are counted manually.
* The data is input on the computer and the salary sheet is calculated through Microsoft Excel.

However, there are many problems in the current system. The attendance register is not a genuine way to record the attendance as it may get lost. Furthermore, every time the salary sheet is calculated all the fields of the sheet, salary, bonus, tax, and allowances etc. need to be input again. Since the attendance register is multi-handed, the staffs sometimes ingeniously mark a fake attendance even they get absent for the previous or upcoming day. The system at present has somehow been a mess which my employee has asked me to clear.

## 1.2.4 Description of the Existing System:

The flow of the current system can be best explained by the flowing flow diagram:



## 1.2.5 Interview 2:

Date: 27th February 2013

Q. What are the fields that you consider while preparing salary sheets for your employees?

A. *I basically provide Basic Salary, Allowances, and Bonus if the company has any profit, Tax and some money deducted for Insurance.*

Q. How do you calculate the salary sheet?

A. *We use Microsoft Excel to calculate the salary sheet.*

Q. How do you retrieve the older records and amend them?

A. *Till now, we have not dealt with those cases frequently. But once, we had done it. We save all the spreadsheet files in a folder in the computer. When needed, we retrieve the records by manually checking all the spreadsheet files which is a bit tedious task.*

Q. What new features do you wish me to add in your system to work at its high performance?

A. *As I already said the previous time, feed the new program with some calculation and note making tools. Design the system in such a way that the operator does not need to work in the computer for as much as now. The new system should also contain search facility, which would enable me to search the required record instantly.*

## 1.2.6 Problems in the Existing System:

After making a cursory observation of the system - How it works, what are the problems, I came to know the lacking in the system exists.

* Manual: The parameters for the system are to be input at every use. This has been a tedious task to do every time. It enables the chances of errors. Data doesn’t have any verification and validation which results future complications if entered incorrectly.
* Prone to cheatings and errors: The attendance is marked present as the signature of the employee. Cheatings on this issue might be seen as anyone can make a fake signature.
* Slow: Since the program is manual, the system is quite slow. The system is difficult to use. The counting of attendance marks and inputting of parameters consume a lot of time. It has been slowing down the efficiency of the system.

## 1.2.7 Alternative Solutions:

There are a number of solutions that can be designed to meet up the requirements. The major ones are mentioned below:

1. Continuation of the existing system with some amendments is also one of the alternatives. The existing system can be pursued by correction of the flaws. However, it would be wise to design a new organized system rather than to update the same old system.
2. Microsoft Access can be a useful application for the system. It is a powerful Database Management System to work with the database management of the attendance and payroll. However, the program has a lot of extra tools that Mr. Paudel is uninterested on. He wants a program to only operate his system with no extra options. Further, He also wants a independent form application with an interactive design that MS Access is incapable of. This makes the use of MS Access as a part of the solution not a good choice.
3. Microsoft Excel could also be used as a solution to the existing problem. It is mainly used for graphical representation of data in form of graphs and charts. Thus, Microsoft Excel can be used to compare monthly attendance of the employees. It also allows different type of calculations to be performed on records. However, features offered by Microsoft Excel are not enough to solve this problem. It cannot manage the salary sheets production would be disorganized too because the data is stored in cells. Since it cannot solve the major problems of the payroll management, Microsoft Excel will be unsuitable choice.
4. Custom-written software can be designed using high-level programming languages such as Visual C#. Programming routines can be coded to meet up the requirements of the new system. A fully organized program with much effectiveness would be designed in this case. Interactive interface, effectiveness, flexibility and swift performance are the characteristics of such programs. With this program, it would be easier to handle with data using the SQL Server and it would be convenient and user friendly to work with the windows form application. A programs designed in C# would be precise and would not contain any extra tools that may not be in used in the system. All these features engendered C# to be an ideal application for the problem.

## 1.2.8 Proposed Solution:

After detailed discussion with Mr. Paudel, we finally agreed to custom written software to design the new system. During the discussion, I had proposed all the solutions and Mr. Paudel was pretty much influenced by this option. He reckoned that the system built this way would be more organized and the handling of the data and the system would be more convenient. Further, he wanted a desktop based application with a form based interface. Thus, he was more inclined to this alternative as it would fulfill all his requirements for his new system without any additional unwanted features. With an analysis of his requirement and inclination to the use of custom written software, I planned of using Visual C# and SQL server to design the custom written software for the new system.

## 1.2.9 Requirements Specification:

After conducting various interview sessions, I concluded with the requirements for the system as specified below:

### 1.2.9.1 Functional Requirements

1. To introduce a system with all the parameters included so that the program is not more manual and works automatically with little or no user interference.
2. To design a computerized attendance system so that the payroll is no more susceptible to errors in attendance.
3. To minimize the errors while inputting of data by validation and verification process.
4. To improve the speed of the payroll calculation as much as possible.
5. To introduce forms those are linked to other forms so that data repetition is minimized and the record is automatically updated.
6. To design a user-friendly interface so that there will be no problems for the user working in the system with full flexibility.
7. To provide the output in a form of report after all the calculations are made.

### 1.2.9.2 Input Requirements

|  |  |  |
| --- | --- | --- |
| S.N. | Category | Fields |
| 1. | Employee | Staff ID, Name, Designation, Address, Date Of Birth, Contact Details, Working State |
| 2. | Attendance | Attendance state |
| 3. | Salary | Basic Salary, Allowance, Insurance |
| 4. | Login | Username, Password |

### 1.2.9.3 Output Requirements

|  |  |  |
| --- | --- | --- |
| S.N. | Report | Fields |
| 1. | Daily Attendance | Staff Id, Name, Designation, Address, Contact Number, Attendance State |
| 2. | Monthly Attendance | Staff Id, Name, Designation, Address, Contact Number, Number of days present, Number of days absent |
| 3. | Monthly Payroll | Staff Id, Name, Designation, Number of days present, Basic Salary, Allowance, Insurance, Net Salary |
| 4. | Perspective Pay slip | Staff ID, Name, Designation, Address, Contact Number, Number of days Present, Basic Salary, Allowance, Insurance, Net Salary |

### 1.2.9.4 Software 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

### 1.2.9.5 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.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**Mr. Sudersun Paudel Bishal Regmi**

1. Design

# 2.1 Objectives

In order to come up with an appropriate design for the new system, a brief interaction with Mr. Paudel led me to conclude with following objectives:

1. To design a fully organized system that works without any or little human intervention.
2. To introduce a graphical user interface to make the program user-friendly and convenient to use.
3. To introduce search facility to view the staff details and perspective salary sheets.
4. To avoid redundancy of data.
5. To use Input forms to enter record and output forms or reports to display the salary sheet.
6. To minimize errors while record is being entered to the system.

With this specification, a lucid framework for the new system struck into my mind. Mr. Paudel and I have agreed upon these objectives and I am willing to start this project considering the facts that the goals are attainable.

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

**Mr. Sudersun Paudel Bishal Regmi**

# Nature of the Solution

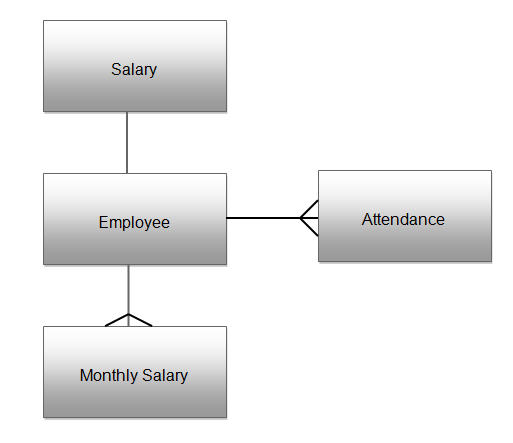
## Proposed File Record and Data Structure

The proposed data structures are listed below. The tables are completely normalized. The tables are split into smaller units as to avoid data redundancy and repetition. The tables have link in order to avoid data inconsistency. The efficiency of the system and size of the files are highly considered.

* Login Table
* Employee Table
* Attendance Table
* Salary Table
* Monthly Salary Table

## Detailed Systems Design

After finalizing with the objectives, the process of design for the solution started. A rough interpretation of the problem engendered some entities for the new system. The Entity Relationship Diagram for the proposed system is shown below.



## Tables

### 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.

## Input Design

### Login Form

Enter the values for the labels in their corresponding field boxes.



Click this button to cancel and close the form

Click this button to login to the system

This form deals with the login form. The form is used to check if the user is authorized to use the system or not. The username and the password are input before getting access to the system. When the login is button is clicked, the input username and password are checked in the table from the database and under matching of the credentials the user gains access to the system.

### Add Employee Form

Enter the respective details for the tables on their corresponding field boxes



Click this button to cancel the operation and close the form

Click this button to add details about new employee

This form deals with the employee table. The form adds new employee record to the table. The basic information about the employee is get through this form. The name label has three text boxes that represent First, Middle and Surname respectively. Similarly, the other fields are to be filled with their corresponding labels. Date of birth is to be selected from the date time picker. If the employee is attending the office regularly without failure, the working state should be active. If the employee is sporadic in the office, he is inactive. Clicking the add button adds the new employee record to the employee table. On clicking of cancel button, the operation is cancelled and the form is closed.

### Edit Employee Form

Enter the respective data for the labels in the corresponding field boxes



Click this button to cancel and exit the form

Click this button to delete the current employee record.

Click this button to update the details about the current employee

This form also corresponds to the employee table. This form is used when the details about the employee is to be updated. The primary function of this form is to update the employee’s basic information or delete the employee. All the values for the labels are retrieved from the employee table. The employee Id and name are not allowed to be edited. Other information can be edited. The employee whose information are to updated is selected from the drop down box and his corresponding information display automatically on the respective filed boxes. If the employee has left the office, the radio button with label “left” is checked. When the update button is clicked, the respective field values are updated in the employee table. When delete button is clicked, the respective record is deleted from the table. On clicking of cancel button, the operation is cancelled and the form closes.

### Salary Parameters Form

# 

Enter the respective data for the labels in the corresponding field boxes. The field Employee Id, Last Name and Designation are Read Only.

Click this button to update the salary parameters for the selected employee

The salary parameter form corresponds to the salary table. It is used to update the salary parameters for the respective employee. When a new employee is added, all the values for basic salary, allowance and insurance are stores as “0”. The form contains some read-only fields that are retrieved from employee table such as Employee Id, Employee Last Name, and Designation. The employee for whose data is to be operated is selected from the drop down box. The basic salary, allowance and insurance field boxes display the corresponding values from the salary table which can be edited. On clicking on the update button, the new values for the fields are updated in the salary table.

### Attendance Form



Select the attendance for the selected employee

Select the employee for whom the attendance is to be marked

Select the day for the attendance to be marked

Click this button to add the attendance details of the selected employee for the selected date

This form corresponds to the attendance table in the database. This form is used to update the daily attendance of the employee. The date for when the attendance is to be marked is selected from the date time picker. The employee whose attendance is to be marked is selected from the drop down box. The value for the employee name, employee Id and designation is retrieved from the employee table for different values of the Employee name drop down box. The Attendance drop down box has two values – Present and Absent. The attendance of the employee on the respective day is selected. On clicking the done button, the respective values for the employee attendance are added in the attendance table.

## Output Design

### Employee Report



The Employee report retrieves all the values in the “employee” table and displays them in the data grid view. The boolean values are represented by a tick meaning “True” and blank meaning “False”.

### Monthly Attendance Report



The monthly attendance report retrieves the attendace details. It uses the values of month and year from the combobox as selected by the user and displays the respective results for the columns in the data gridview from the attendance table.

### Monthly Salary Report



The monthly salary report displays the calculated monthly salary of the employees for the selected date. The user selects the month and year from the respective combobox. The respective values of employee details, present days, salary parameters are retrieved from the employee, salary and attendance table and monthly salary and net salary are calculated using simple formulae. The values are displayed in the data grid view under their respective column headers.

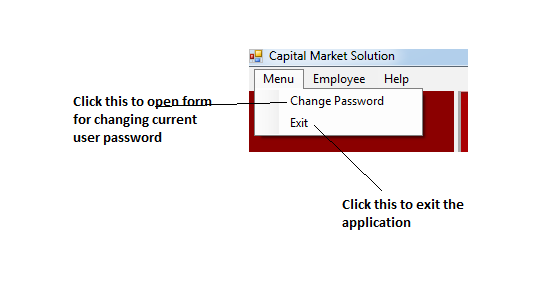
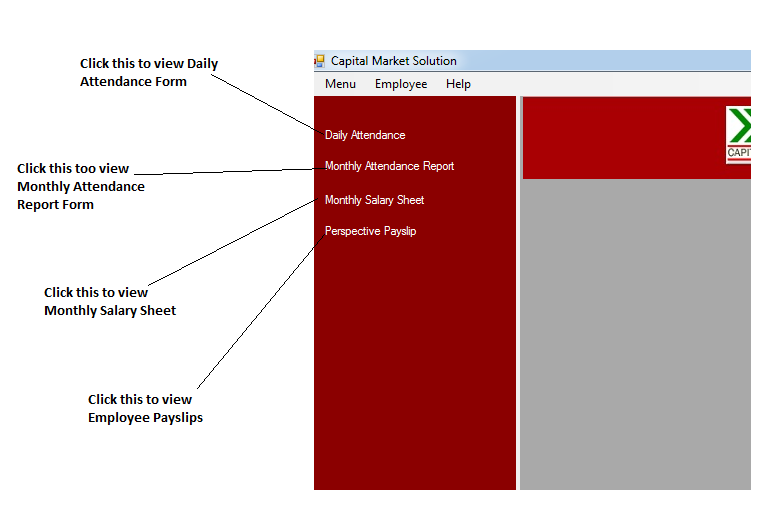
### Employee Payslips Report

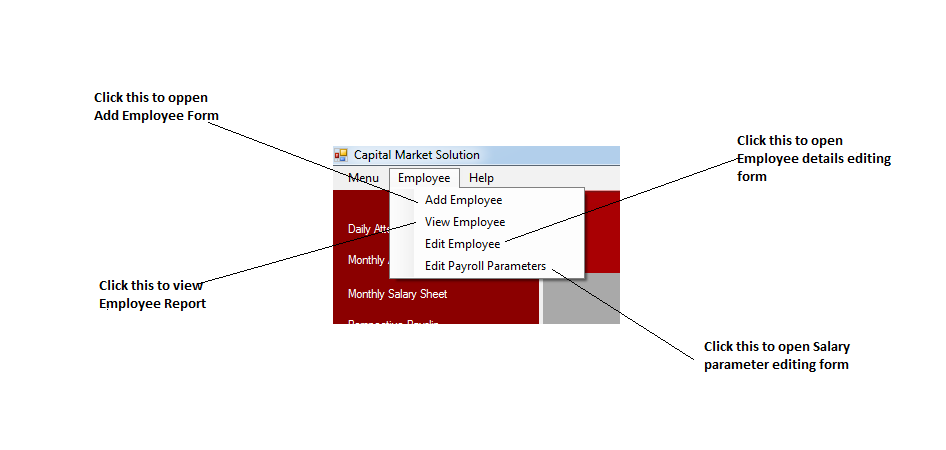


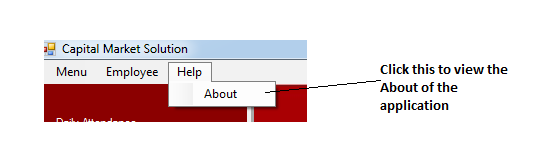
The employee payslip report displays the monthly salary for the selected employee for selected date. The user needs to select month, year and employee name from their respective combo boxes. On clicking the show button, the query retrieves all the values for the fields from the “monthlySalary” table.

## Menu Design

The menus only exist in the main window form and the User Controls have been used to displays some of the forms whereas some forms are displayed as normal windows forms.







## 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.

# 2.3 Intended Benefits

* **Easy to use and user friendly**: The new system has a menu and form based interfaces which will allow the user to operate the system easily. Basic file operations can be just done on a few clicks.
* **Cost benefit:** Though the capital investment cost is high the new system will prove economic in the long run. The computerised system would mean that there will be minimal use of paper eliminating the cost of designing and printing the documents and the stationery. The computer based system also eliminates the need of hiring an accountant to maintain and analyse the documents.
* **Security of data:** The new system will record data in electronic form in the database. The facility of backup ensures that data is recoverable in case of loss or corruption. The password protection of the software ensures that right person accesses the right data and unauthorised access is prevented. Also, the old filing system is replaced thus the problem of file being damaged by rainwater and rats will be solved
* **Minimization of errors:** Since the new system will be a automated computerized system, the probabilities of human made errors would be removed thus increasing efficiency of working. The validation amd verification processes are intended to minimize the errors as much as possible.
* **Speed of operation:** Unlike the previous system, the system will work much faster as the parameters should not be entered at every operation. Further, the attendance marks need not to be counted as they would be stored in the database.

# Limits of the Scope of the Solution

## 2.4.1 System Limitations

Despite of the features of the new system design, there are a few slips and limitation in the system. Some of them are listed below:

* The application makes a great deal of use of memory resources and needs a powerful computer to run.
* The attendance system is not automated as it is to be marked manually daily.
* The payslips for the employee cannot be printed.
* The reports view cannot be changed.
* New users cannot be added. Only a single user is allowed to use the system.

## 2.4.2 File Size Estimation

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Table Name | Maximum size of record in table (bytes) | Estimated Number of records per year | Maximum Size of Table | Maximum size of table with overhead (10%) |
| Login | 29 | 1 | 29 bytes | 29 bytes |
| Employee | 364 | 25 | 9100 bytes  = 8.9 KB | 10010 bytes  = 9.8 KB |
| Attendance | 17 | 7925 | 134725 bytes  = 119.1 KB | 135518 bytes  = 132.3 KB |
| Salary | 32 | 25 | 800 bytes  = 0.8 KB | 880 bytes  = 0.9 KB |
| Monthly Salary  Maximum size of database = 0.02 + 9.8 + 132.3 + 0.9 + 15.5 = 158.5 KB | 48 | 300 | 14400 bytes  = 14.1 KB | 15840 bytes  = 15.5 KB |

1. **Software Development, Testing & Implementation**

# Software Development

## 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**();**

**}**

**}**

**}**

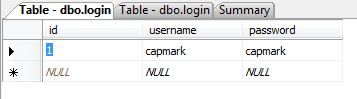
## Development of Database

### Login Table

Design View

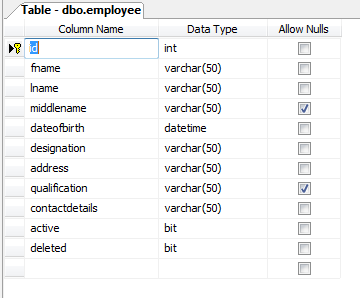


Data Sheet View

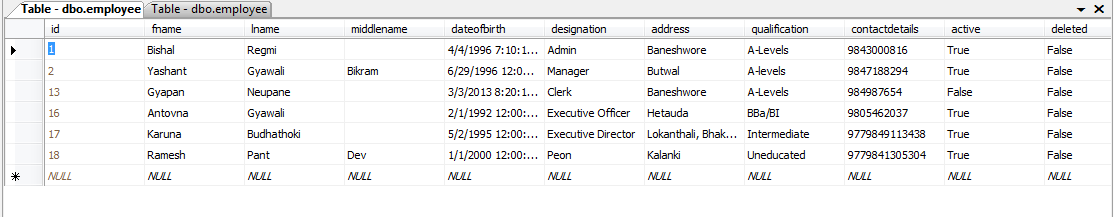


### Employee Table

Design View

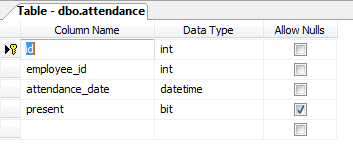


Data Sheet View

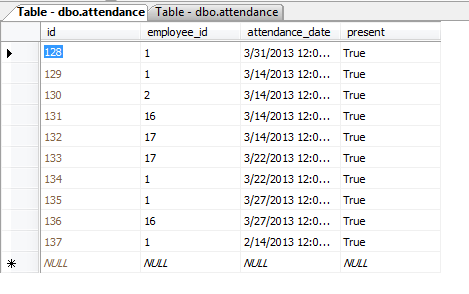


### Attendance Table

Design View

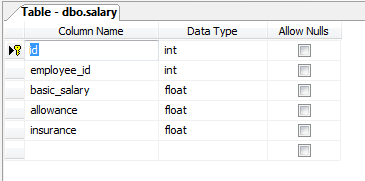


Data Sheet View

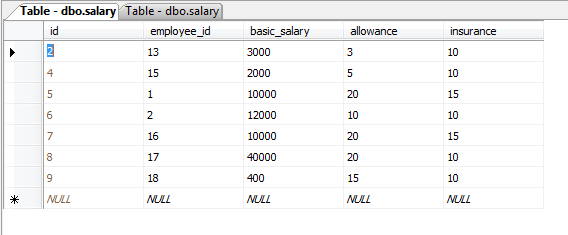


### Salary Table

Design View

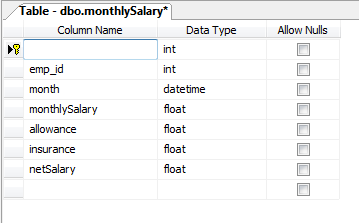


Data Sheet View

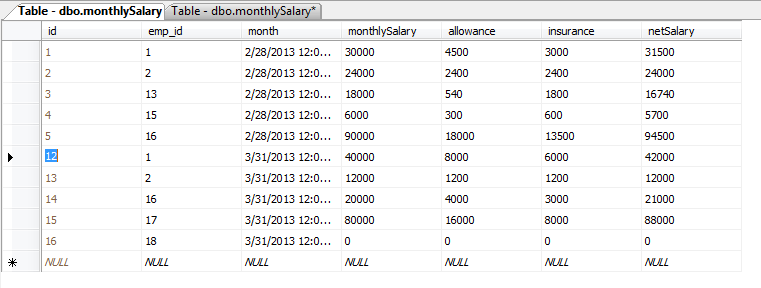


### Monthly Salary Table

Design View



Data Sheet View



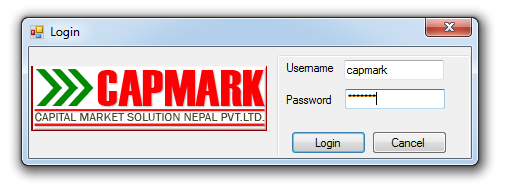
## Development of Input Methods

### Login Form

Design View



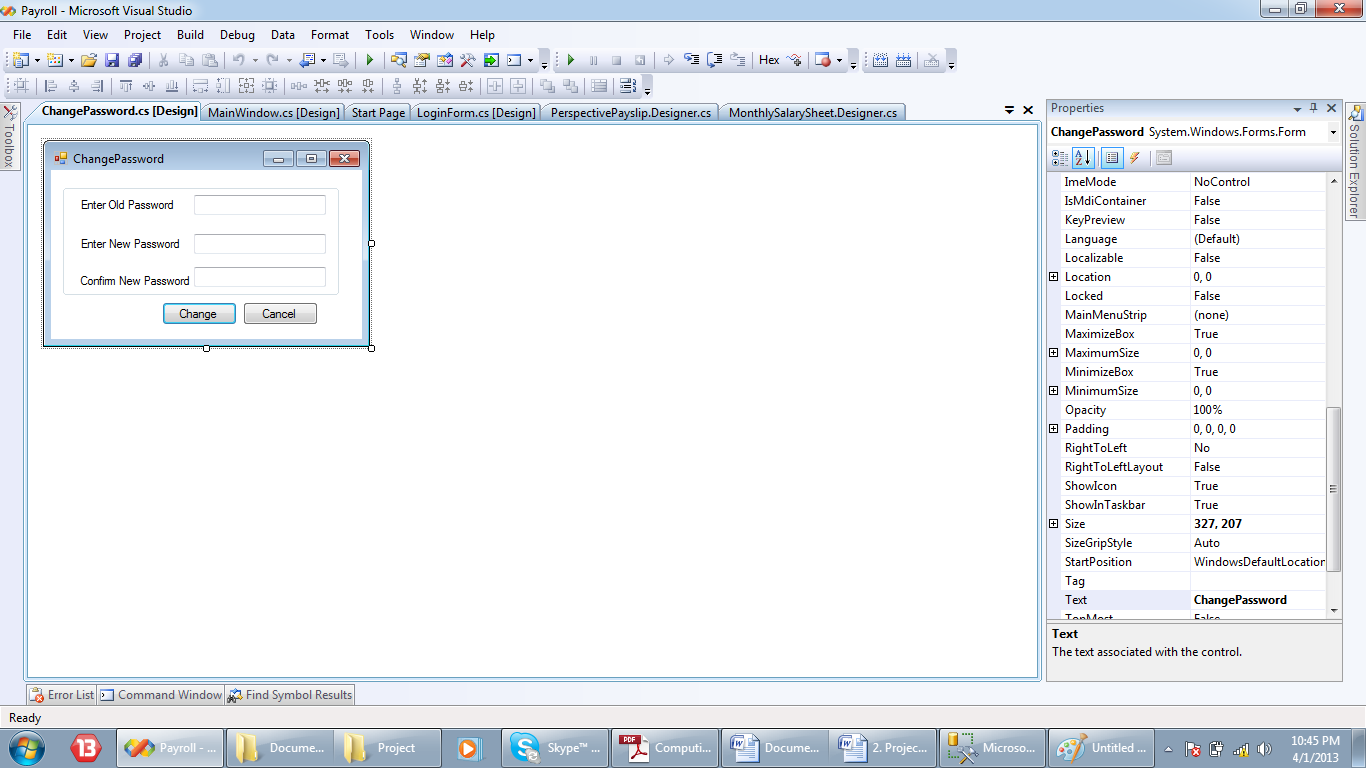
Run View



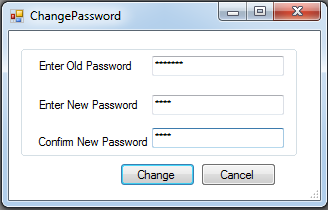
* Username is input in the username textbox.
* Password is input in the password textbox.
* The login button directs the system to the main window after the authentication of username and password.
* The close button and cancel button closes the form and exits the application

### Change Password Form

Design View



Run View



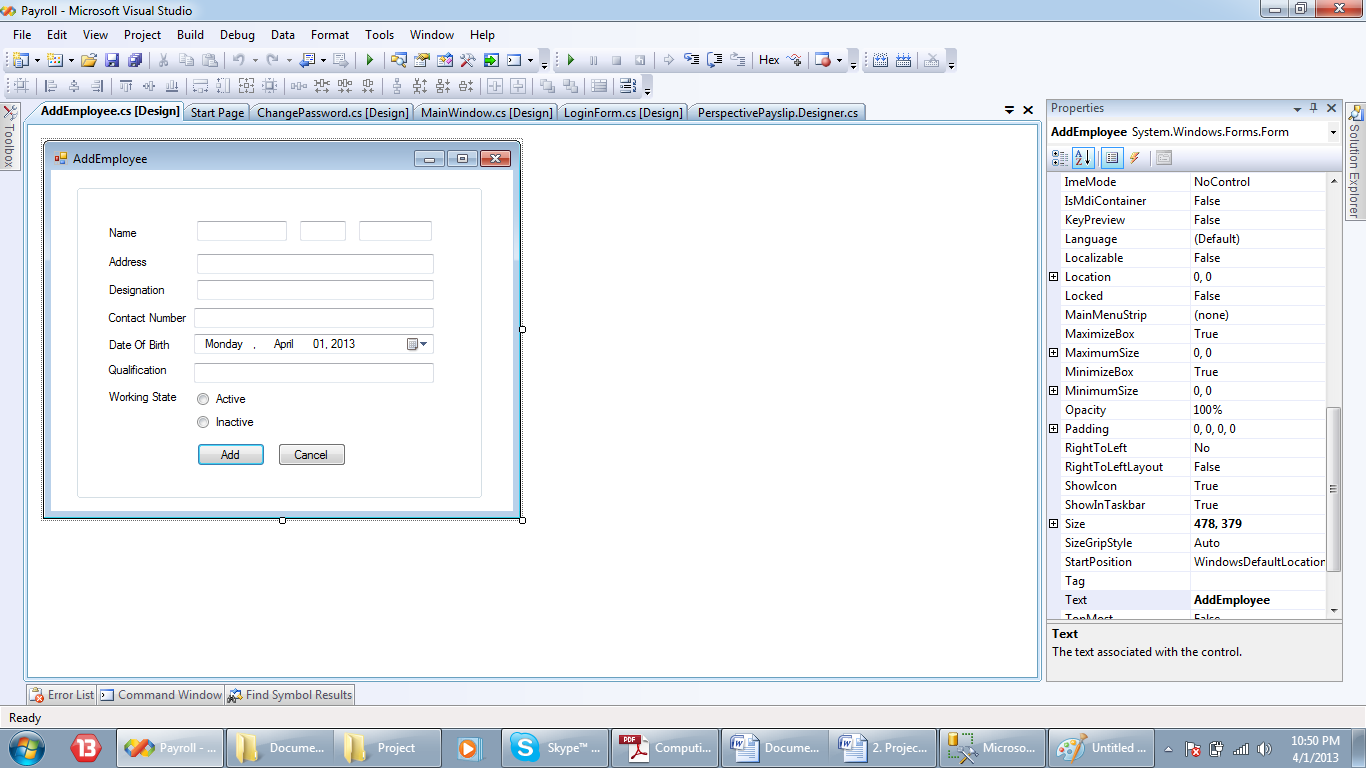
Enter the corresponding values for labels in their field boxes

Click this button to cancel and exit the form

Click this button to update the new password

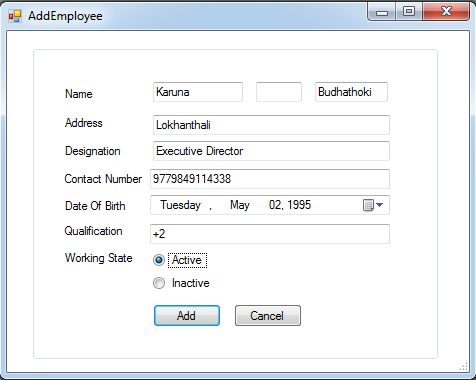
### Add Employee Form

Design View



Run View

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.

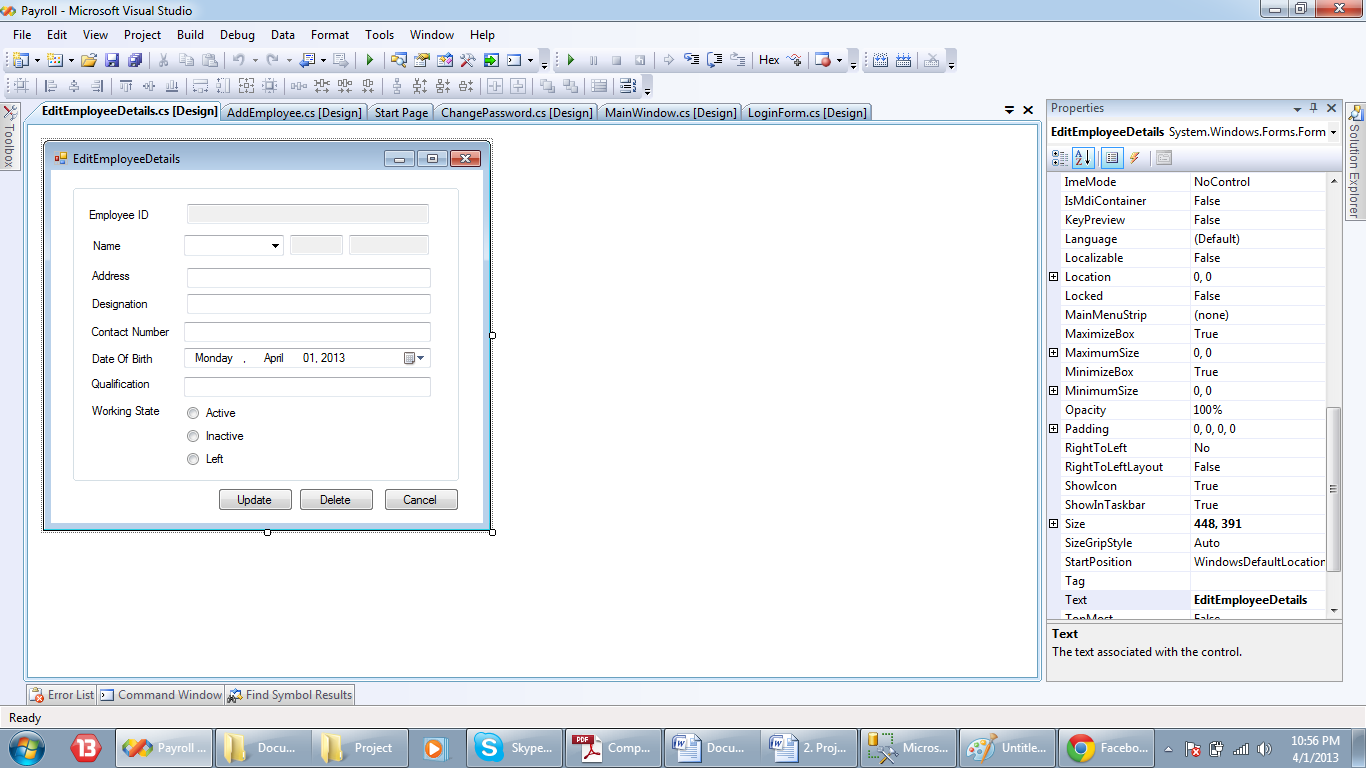


Click this button to cancel and exit the form.

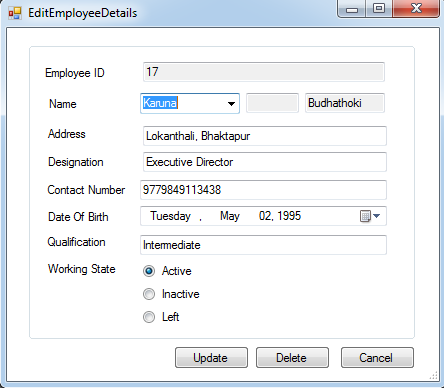
Click this button to add new employee record to the database

### Edit Employee Form

Design View



Run View



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.

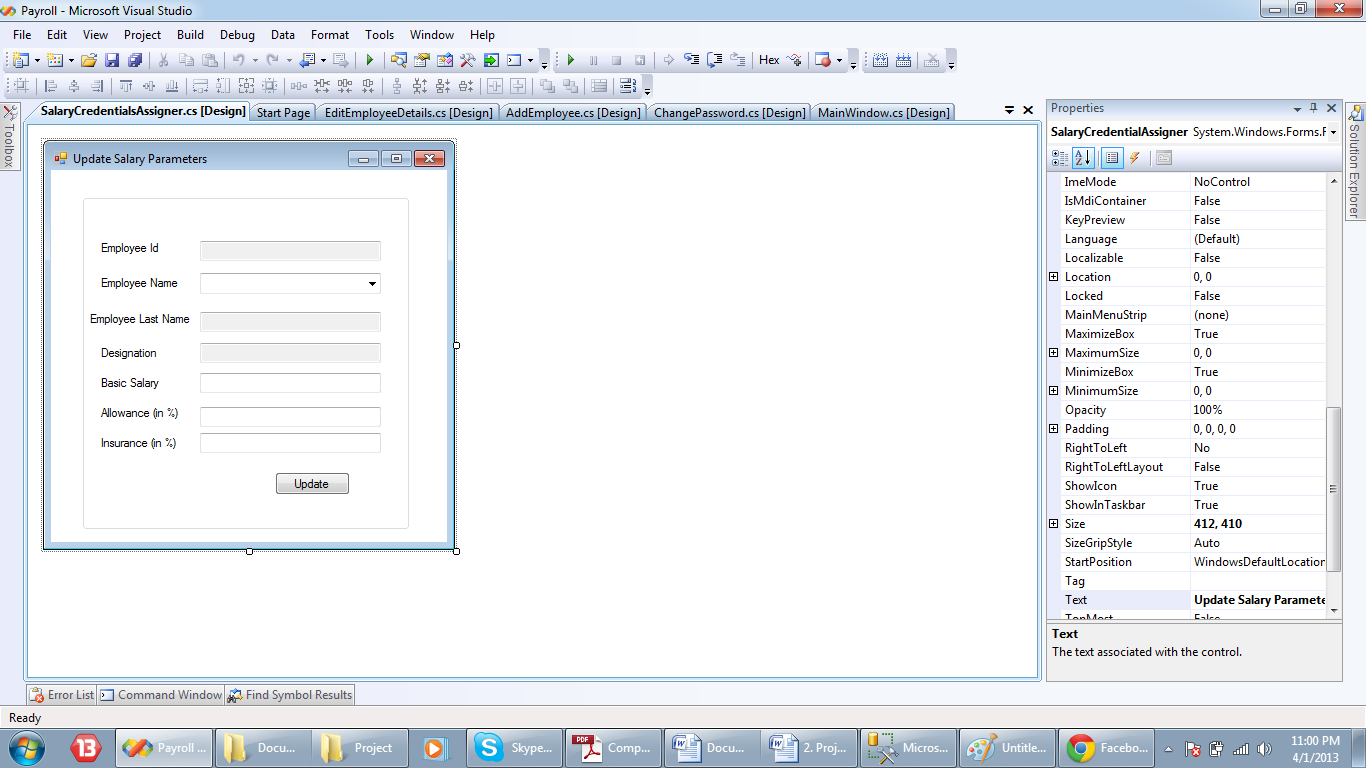
Click this button to cancel and exit the form

Click this button to delete the employee from the database.

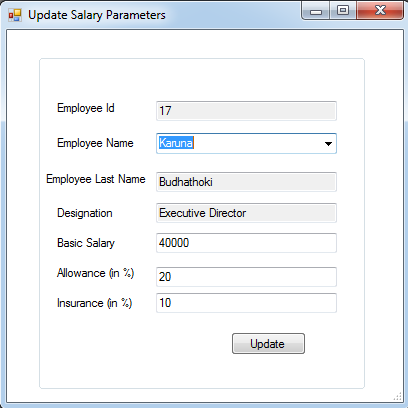
Click this button to update the detail for the selected employee

### Update Salary Parameters Form

Design



Run View

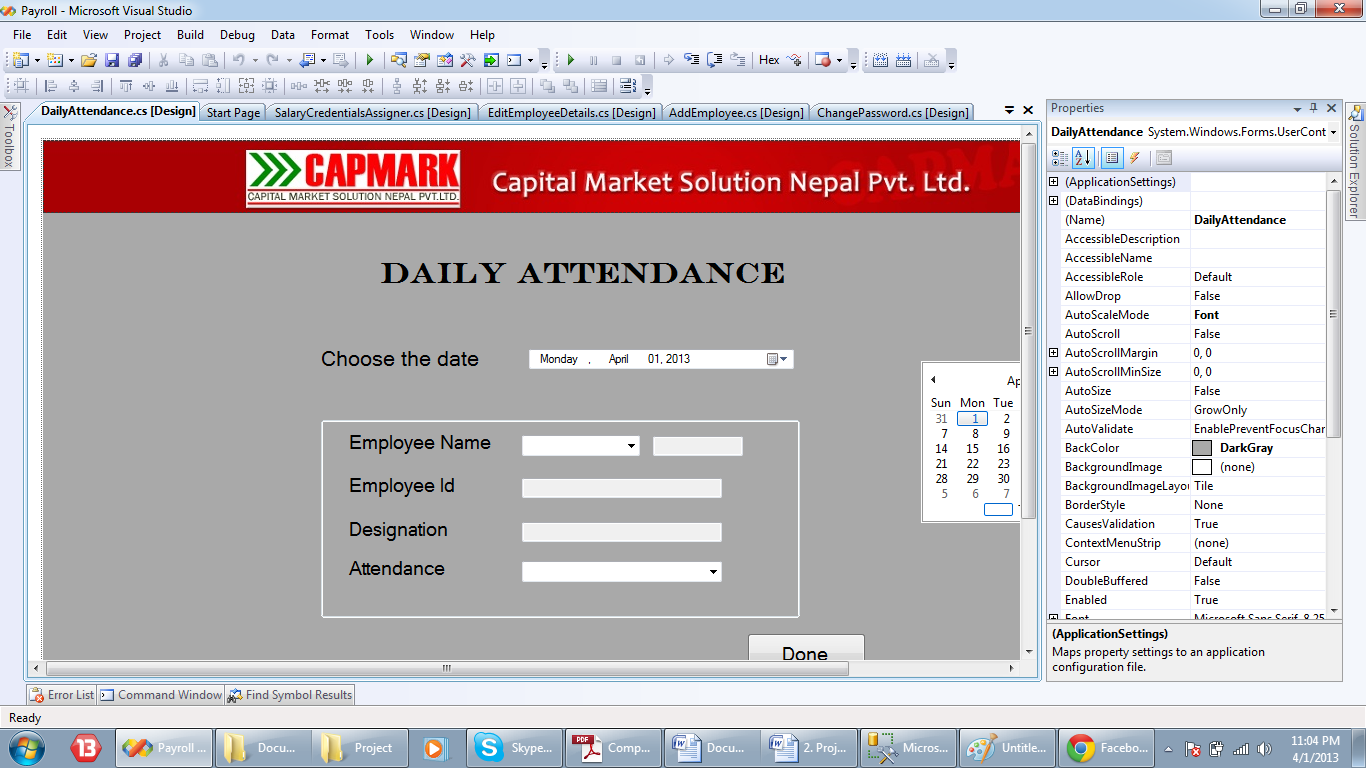


Select an employee for the combobox. To edit, enter the corresponding values for labels in their field boxes.

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

### Daily Attendance

Design View



Run View

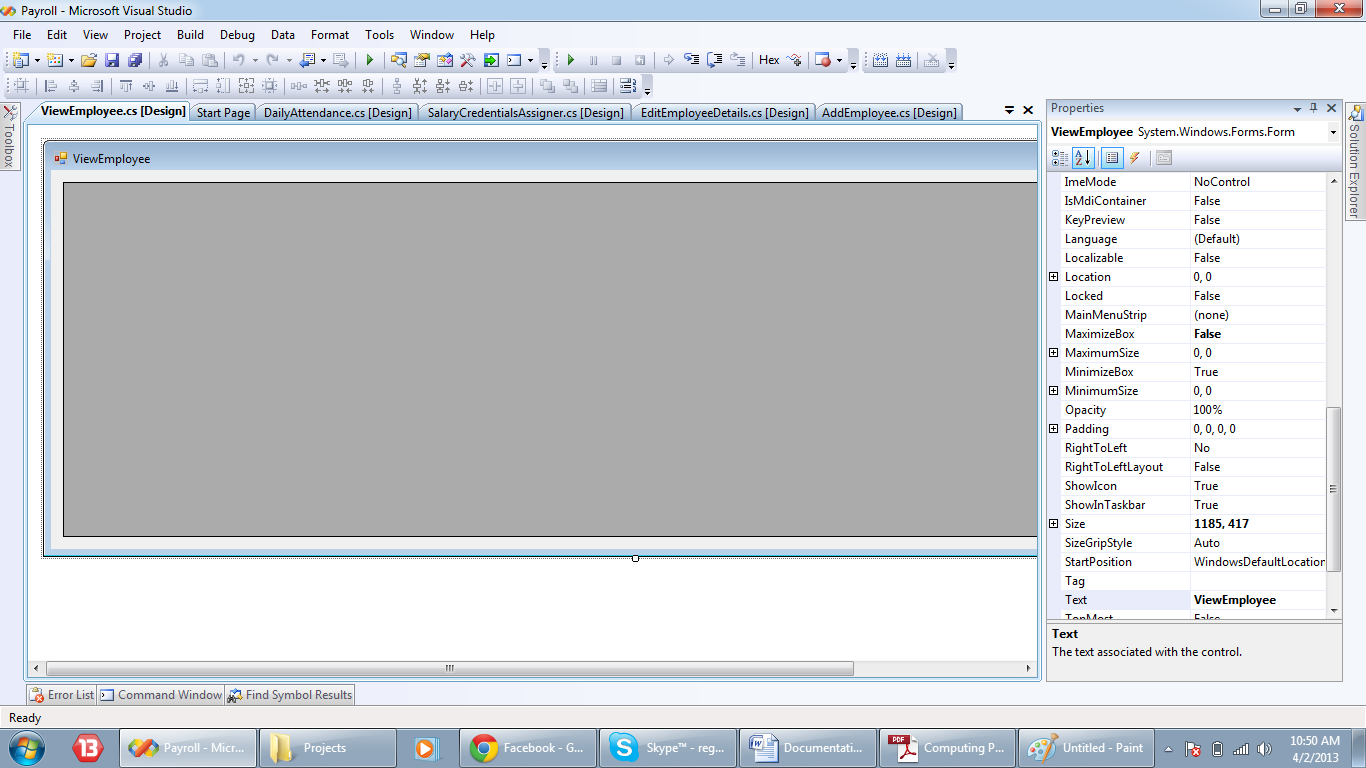


* Select a date from the date time picker for which to mark the attendance.
* Select the employee from the employee name combo box for whom to mark the attendance.
* Select attendance form the attendance combo box (present or false).
* Click “Done” button to add the attendance details about the selected employee to the database.

## Development of Output Methods

### View Employee Report

Design View



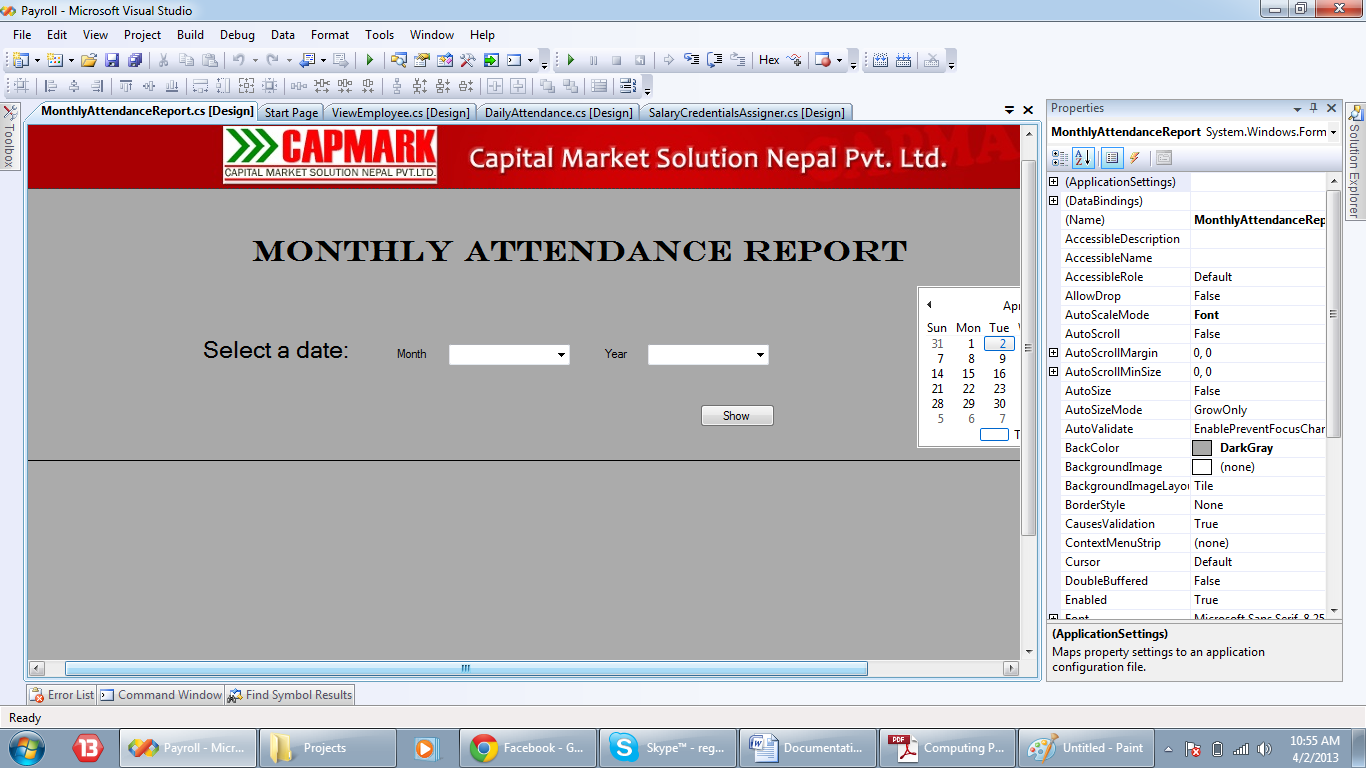
Run View



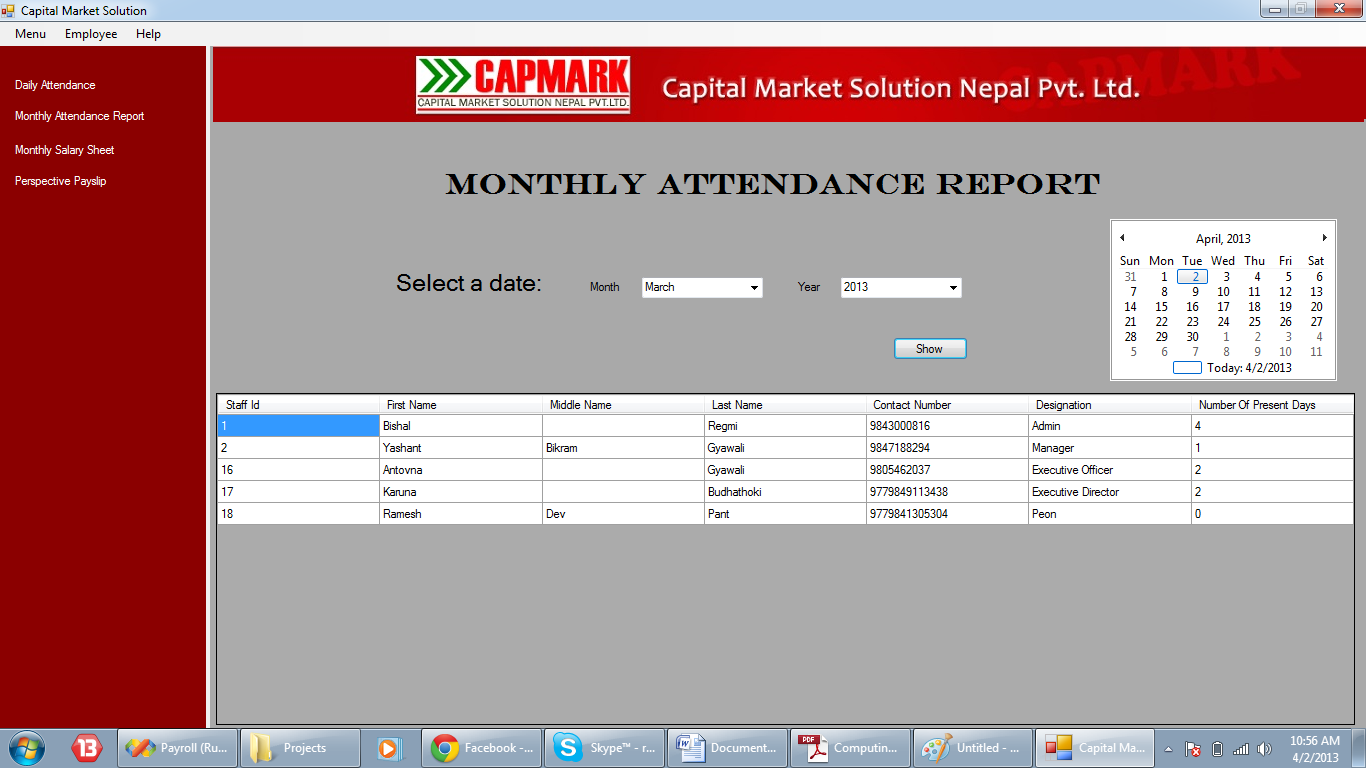
The report shows the employee details about all the employees in a data grid view.

### Monthly Attendance Report User Control

Design View



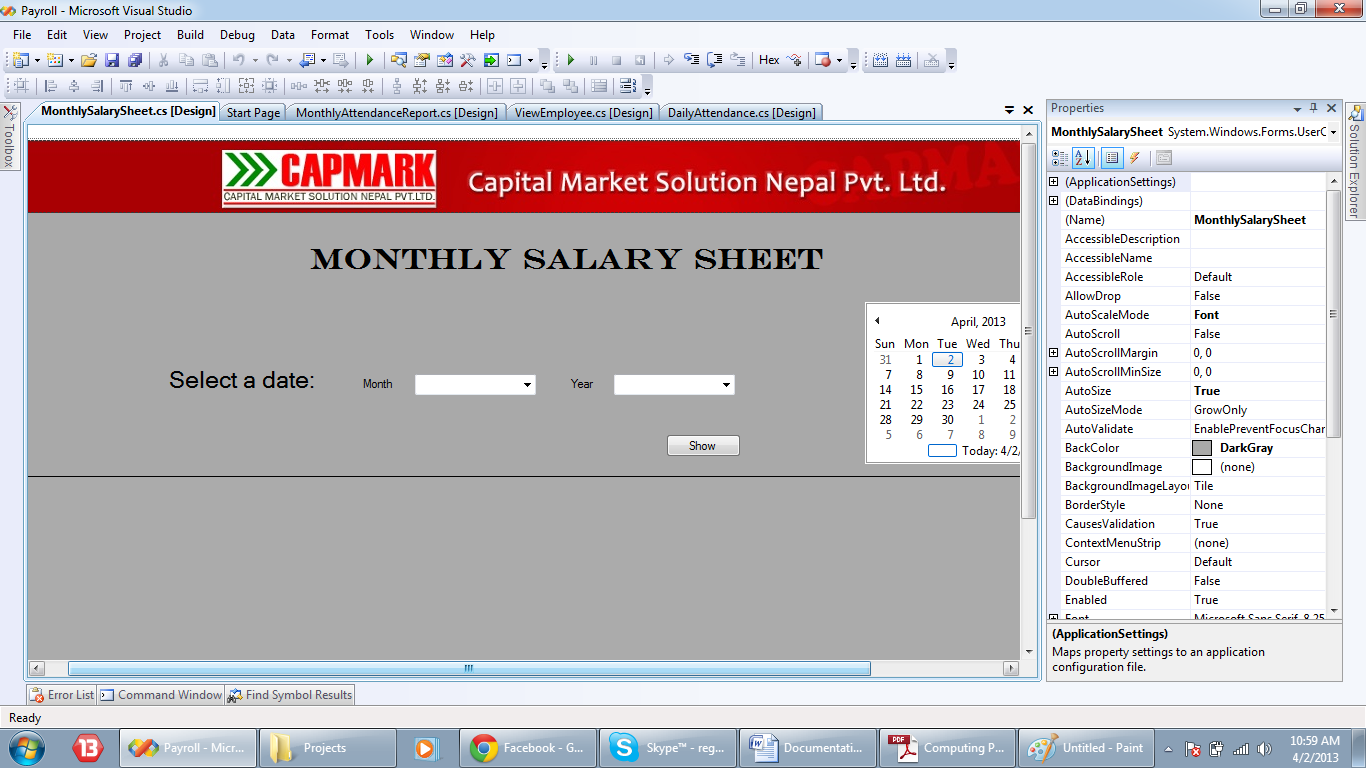
Run View



* 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

Design View



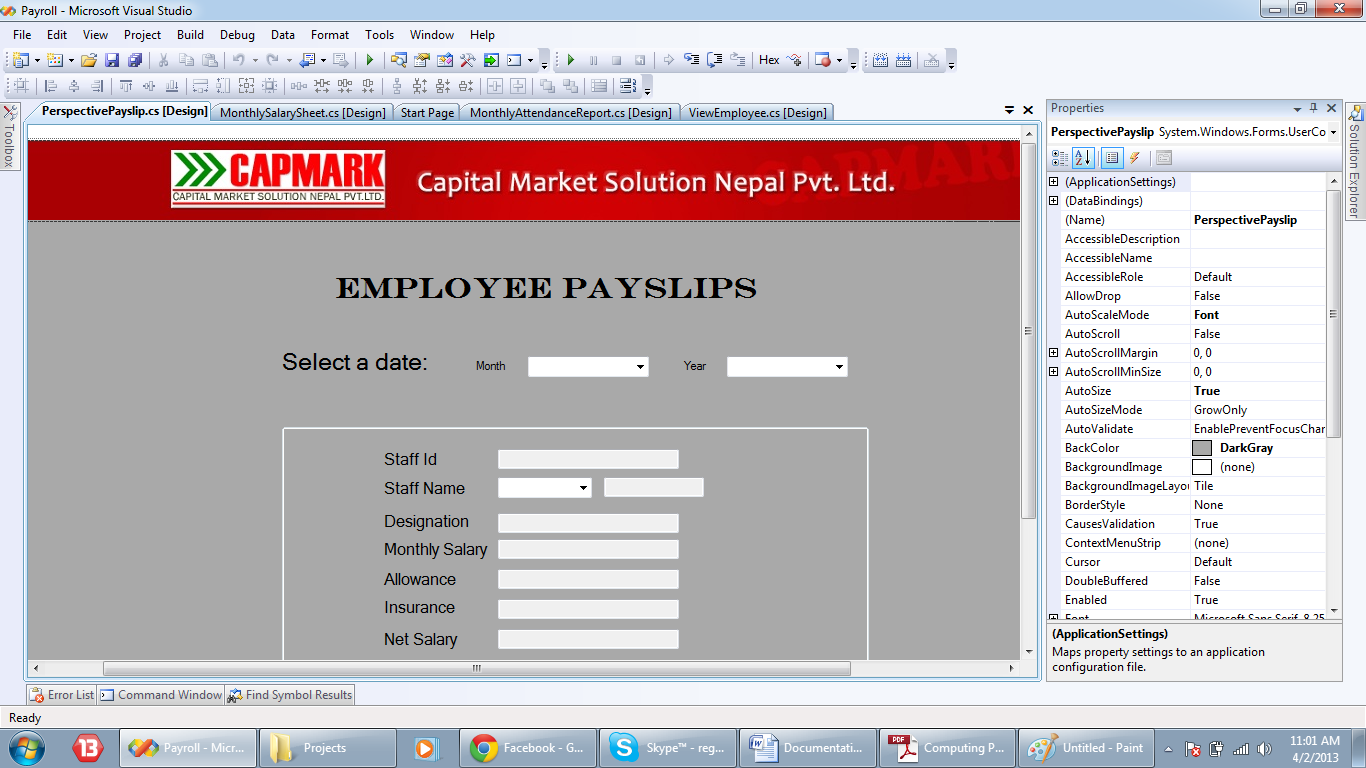
Run View



* 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

Design View



Run View



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.

## Development of Database Queries/Filters

### Query to select username and password from login table

Design view

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

SQL View



### Query to add new employee to the employee table

Design View

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 + "')";

SQL View



### Query to update employee details in the employee table

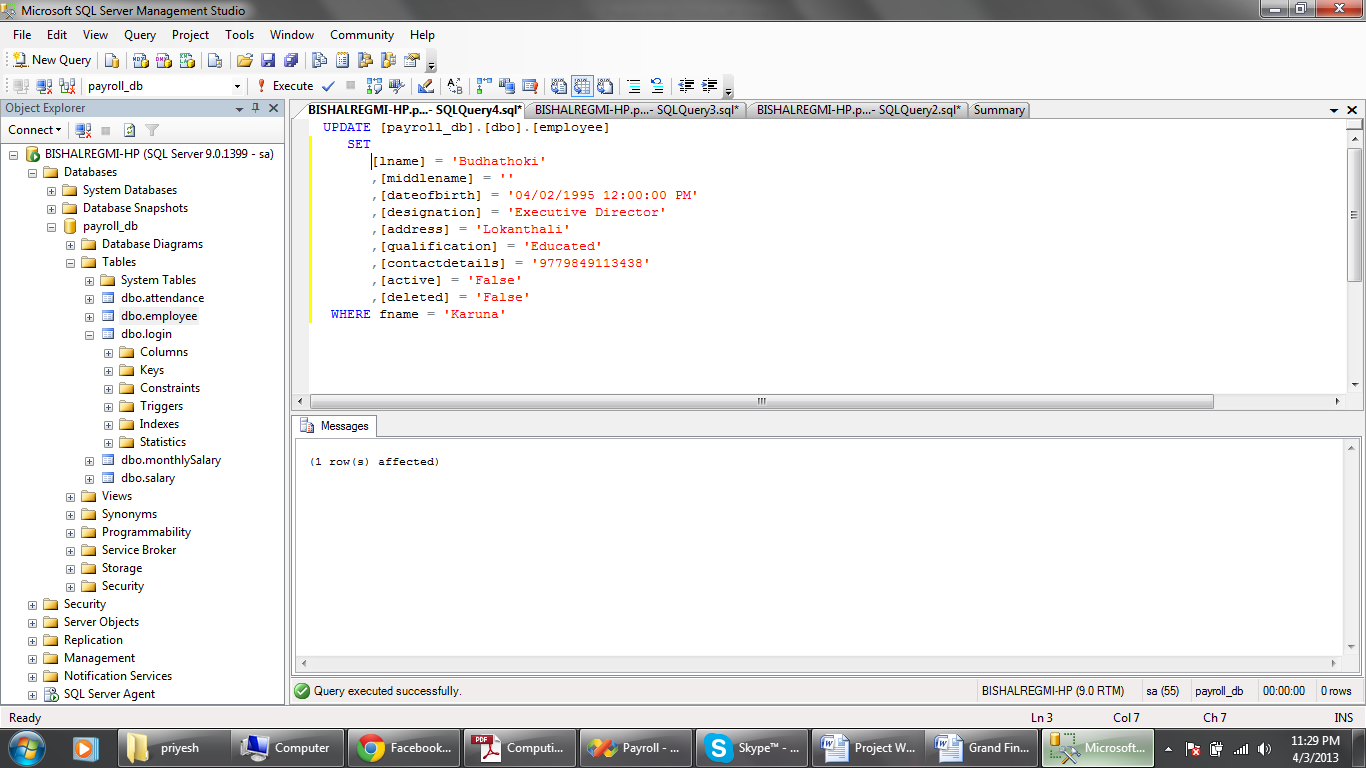
Design View

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;

SQL View

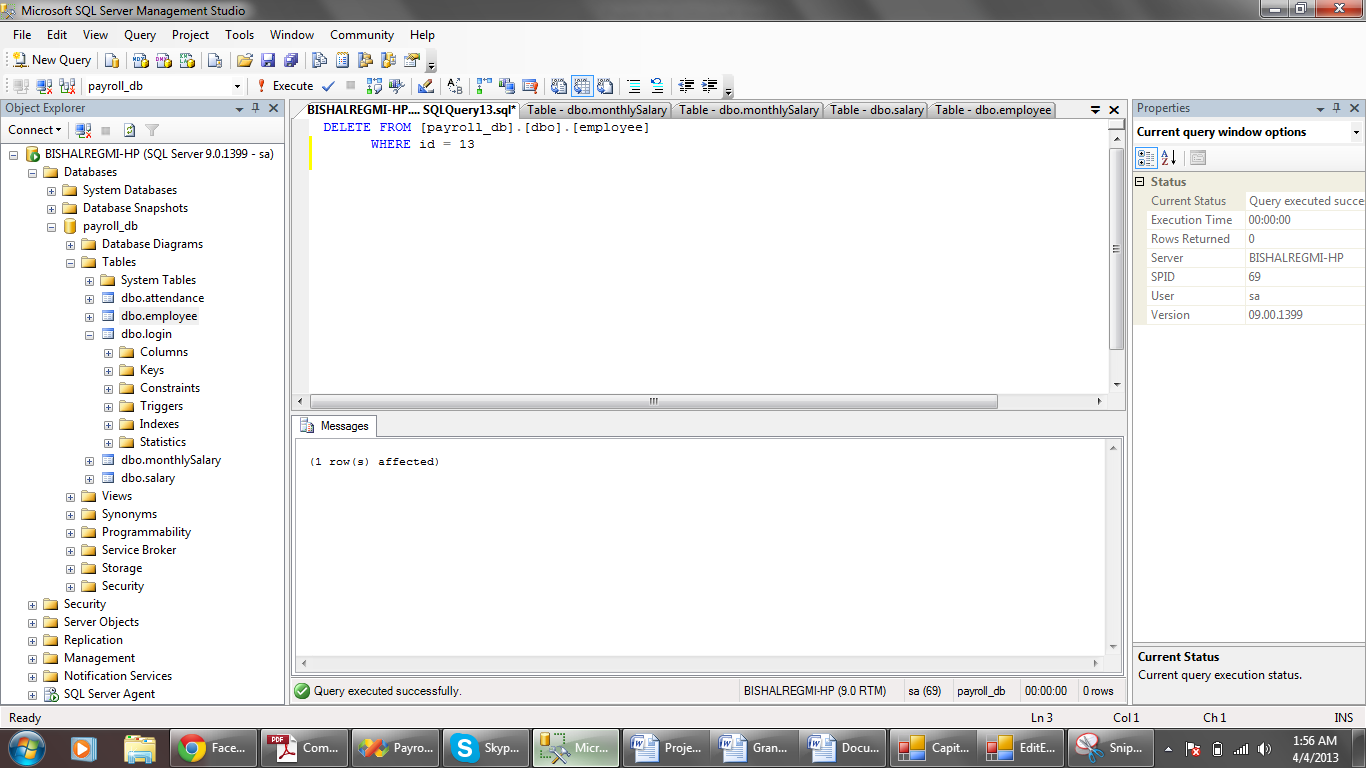


### Query to delete employee from the employee table

Design View

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

SQL View

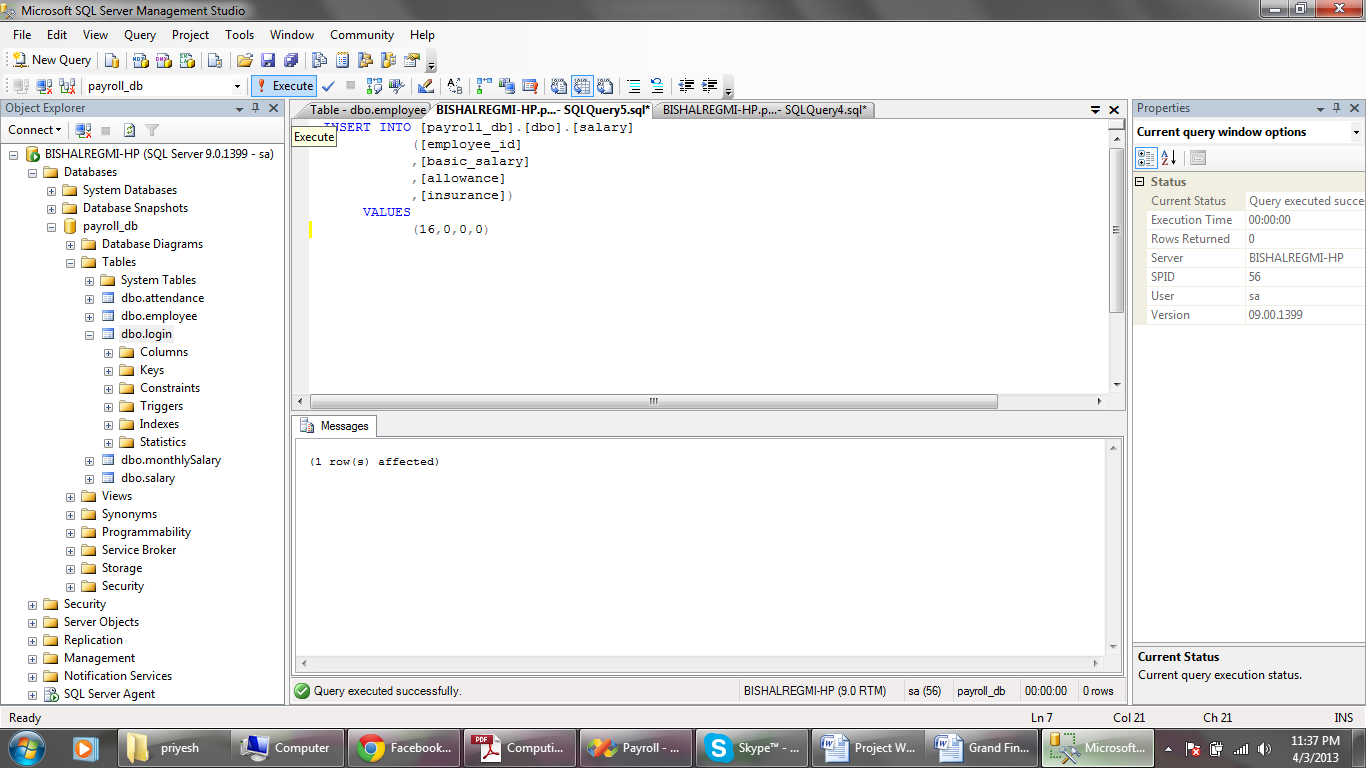


### Query to set salary parameters

Design View

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

SQL View

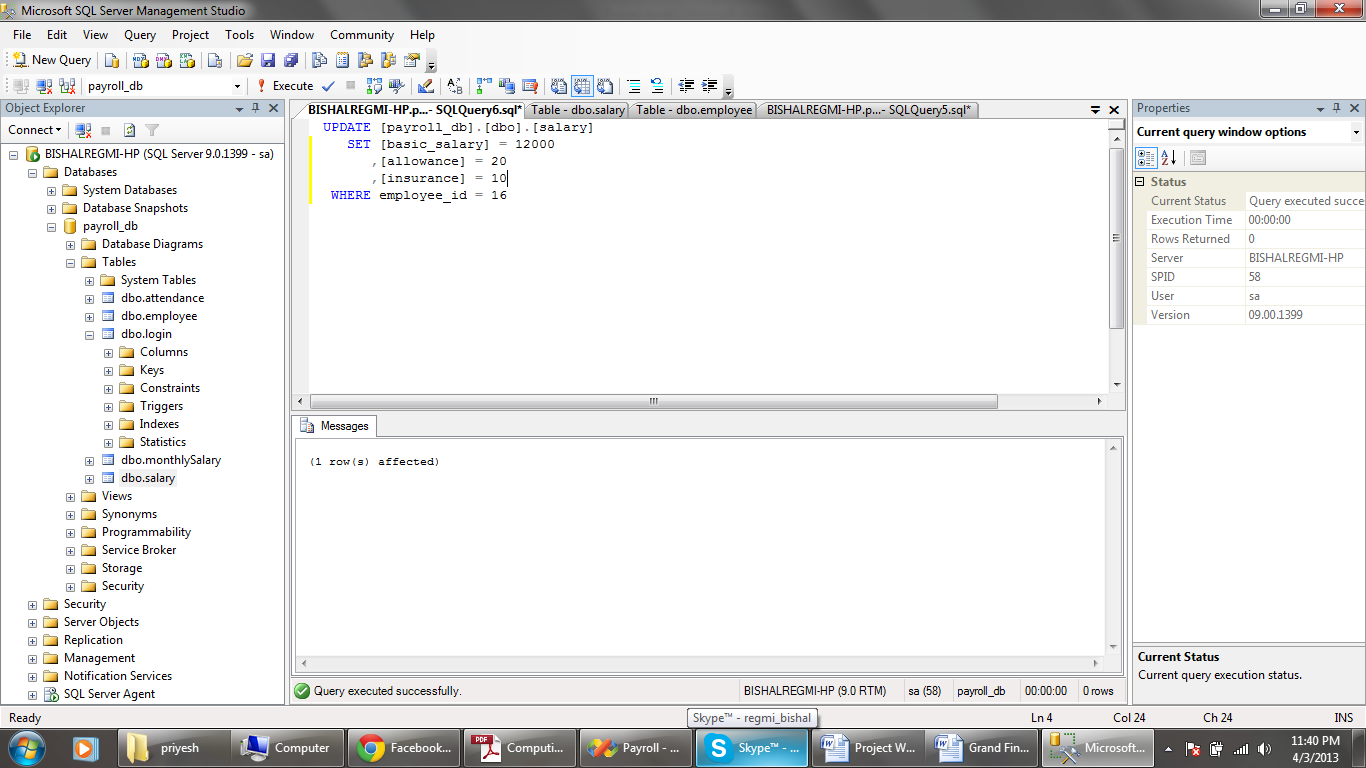


### Query to update salary parameters

Design View

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

SQL View

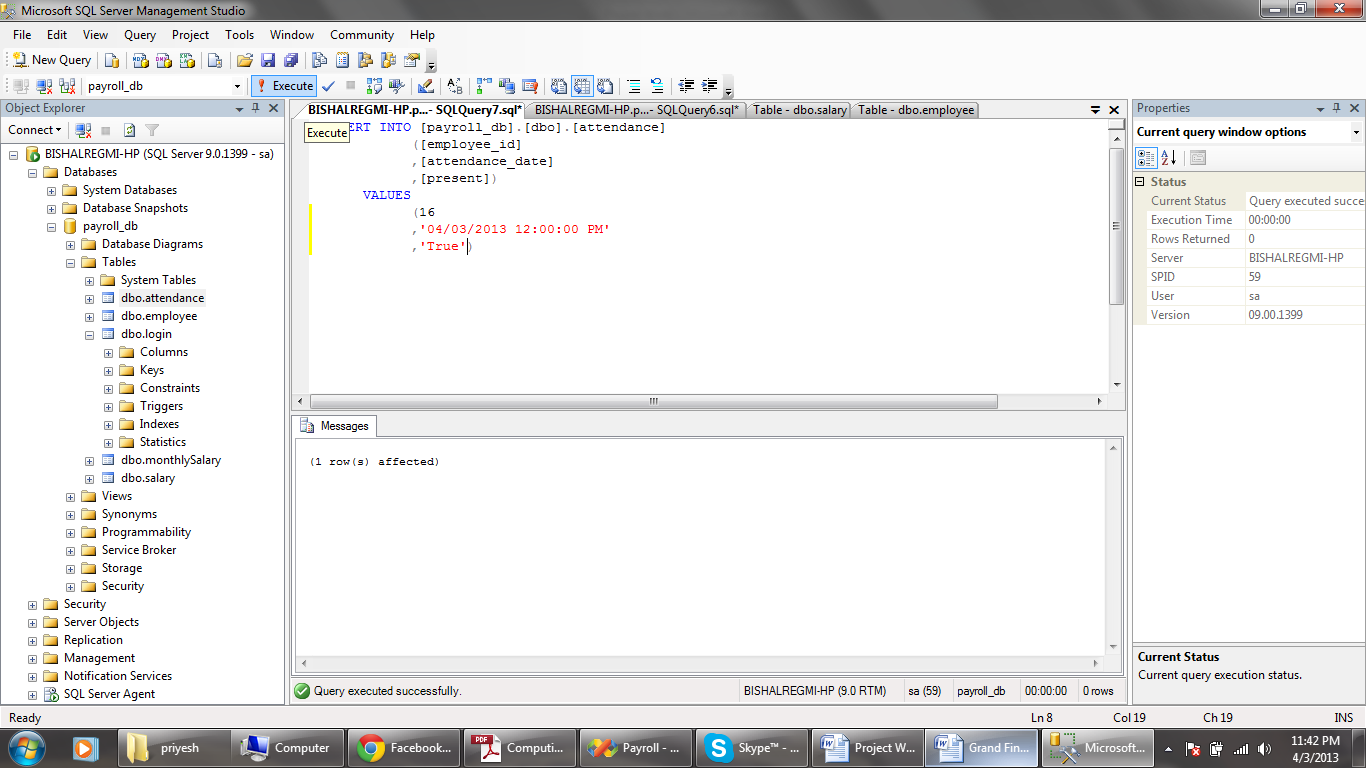


### Query to record daily attendance

Design View

string query = "insert into attendance(employee\_id,attendance\_date,present) values(" + empid + ",'" + dateTimePicker1.Value.Date + "','" + isPresent + "')";

SQL View

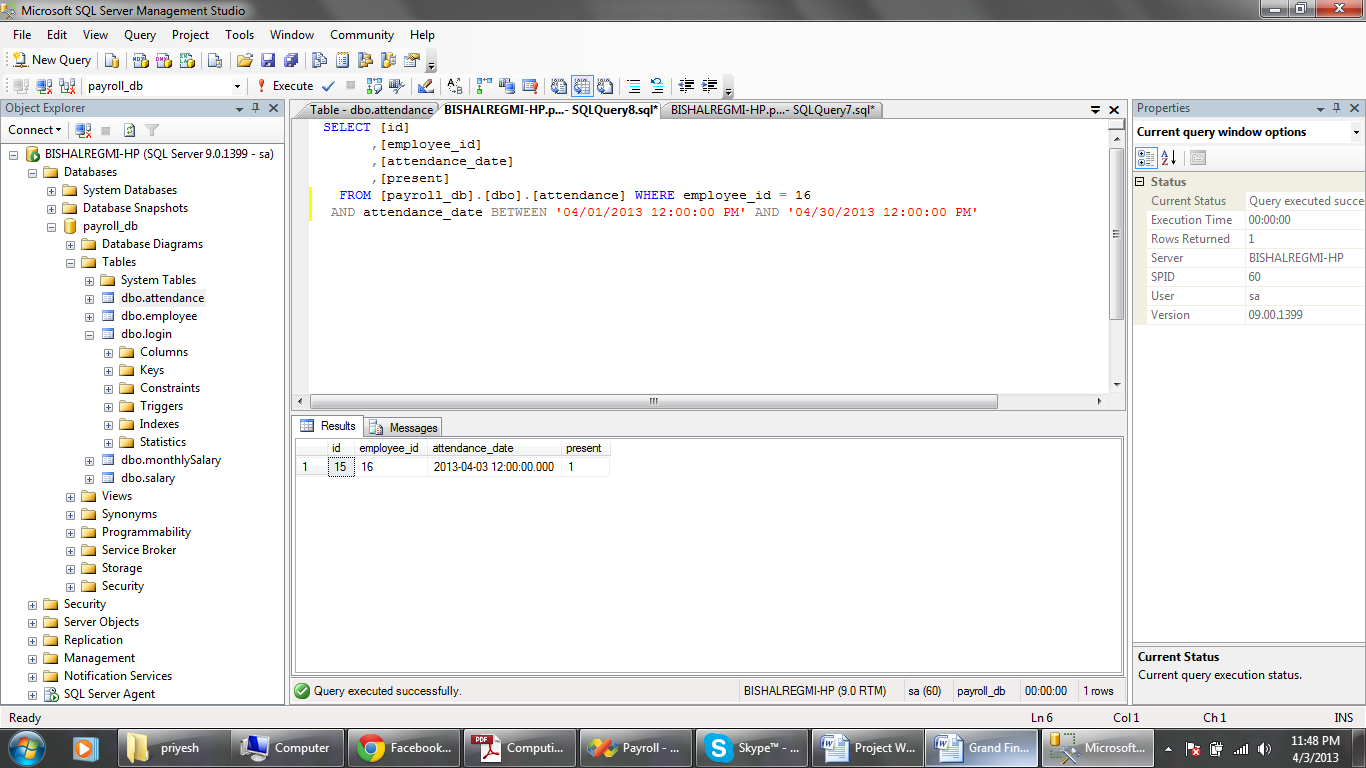


### Query to select attendance details for a given date interval

Design View

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'";

SQL View

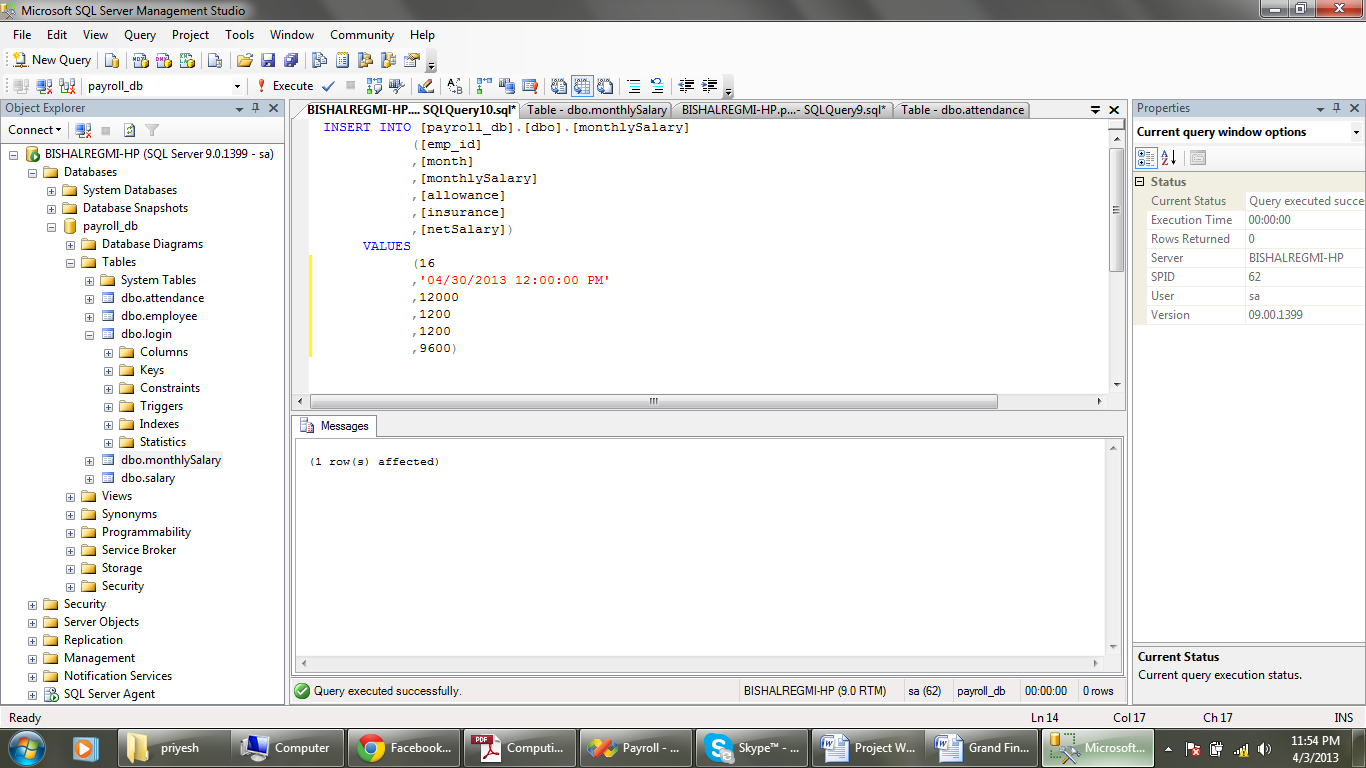


### Query to insert monthly salary

Design View

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 + ")";

SQL View

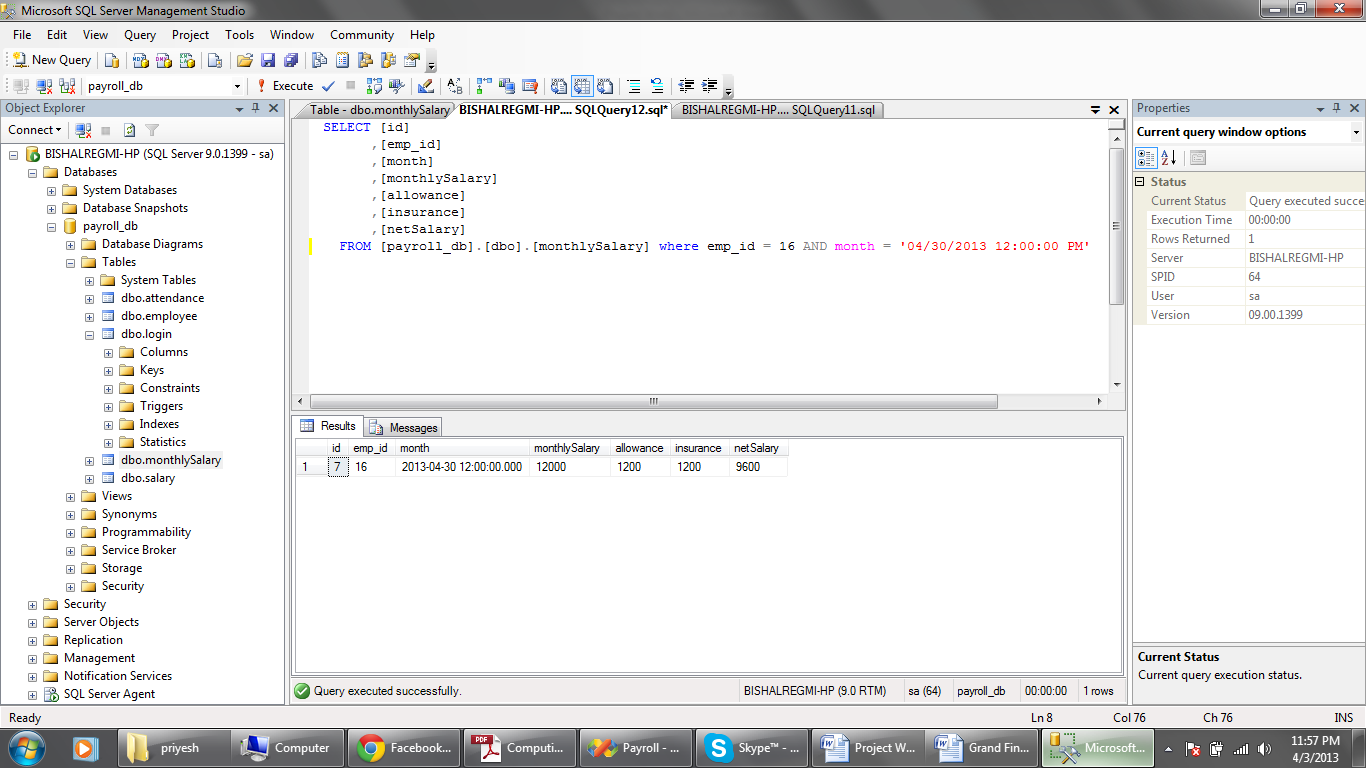


### Query to select monthly salary details from the monthly salary table

Design View

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

SQL View



# Testing

## Testing Strategy

The new system will undergo extensive testing to make sure that program functions as it should. All the feasible paths through the system will be tested.

Alpha testing will be carried by me and this will mainly be the black box testing. The expected outcome will be compared to the actual outcome. If there is any difference between the two outcomes, the problem will be analyzed and investigated, and suitable solution will be formulated. After the alpha testing, the user will be asked to test the program in real work condition.

Normal Valid data, borderline data and invalid data will be input wherever it is appropriate. The normal data is the data which the system accepts without any error. The invalid data should not be accepted by the system and should generate error message. Borderline data are the data which are out of range of the accepted values.

The main concern will be testing whether the records are stored in the database as it should and testing if the data can be retrieved in the form as it should. It will also be made sure that the reports will be generated.

Since the program will be modularized, each screen will be tested one by one.

## Test Plan

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test # | Description | Test Data or Action | Expected Results | Actual Results | Proof Page |
| Testing the login form actions | | | | | |
| 1 | Ensure empty fields are not accepted for username and password | Empty fields | Error Message | As expected | 91 |
| 2 | Ensure the system rejects the incorrect login credentials | Username : abcd Password : abcd | Error Message | As expected | 91 |
| 3 | Ensure the system accepts the correct login credentials | Username : capmark Passowrd : capmark | Login Successful | As expected | 91 |
| Testing the change the password | | | | | |
| 4 | Ensure the empty fields are not accepted | Empty fields | Error Message | As expected | 92 |
| 5 | Ensure the wrong password is not accepted | Old password : hello | Error Message | As Expected | 92 |
| 6 | Ensure the different values for new password and confirm password are rejected | New Password : abcd  Old Password : mnop | Error Message | As Expected | 93 |
| 7 | Test to see thatvalidated values are accepted | Old Password: capmark  New Password : hello  Confirm Password : hello | New password will be updated in the database. | As Expected | 93 |
| Testing the action in Add Employee Form | | | | | |
| 8 | Ensure the empty fields are not accepted | Empty fields | Error Message | As Expected | 94 |
| 9 | Check to see if the valid values are accepted | Name : Karuna Budhathoki  Address : Bhaktapur  Designation : Executive Director  Contact Number : 9779849113438  Date Of Birth :05/02/1995 12:00:00 PM  Qualification : Intermediate  Active : Checked  Inactive :Unchecked | New employee record will be added in the database | As expected | 94 |
| Testing for Editing the Employee Details | | | | | |
| 10 | Ensure that the new values are updated | Address : Lokanthali  Qualification : Educated  Active : Unchecked  Left : Checked | The details about the employee will be updated in the database. | As Expected | 95 |
| Testing for deleting the employee record | | | | | |
| 11 | Check if the employee is deleted on clicking delete button | Clicking the delete button | The employee will be deleted from the database | As Expected | 96 |
| Testing for assigning of salary parameters | | | | | |
| 12 | Test to see if Basic Salary field does not accept other character than numbers | Abcd | Error | As Expected | 96 |
| 1000 | The values are accepted by validation check | As expected | 97 |
| 13 | Ensure that the allowance and insurance fields do not allow numeric values more than 100 | 1000 | Error | As Expected | 97 |
| 20 | The values are accepted by validation check | As Expected | 98 |
| 14 | Ensure that the values are updated in the database | Click the update button | The values are successfully updated in the database | As Expected | 98 |
| Testing for marking the daily attendance | | | | | |
| 15 | Test to check if the attendance is allowed for date greater than the current date | 04/04/2020 | Error | As Expected | 98 |
| 16 | Test to check if the attendance field is allowed empty | Null value | Error | As Expected | 99 |
| 17 | Test to check if the valid data are updated or not | Employee Name : Bishal  Date: 04/04/2013  Attendance : Present | The data is successfully added to the database | As Expected | 99 |
| 18 | Test to check if the attendance for the same employee for same date is allowed for more than once | Employee Name : Bishal  Date: 04/04/2013  Attendance : Absent | Error | As Expected | 100 |
| Test to view the monthly attendance report | | | | | |
| 19 | Test to see if the correct report for the selected date is shown or not | April 2013 | The data from the database is retrieved and correctly shown in the data grid view | As expected | 100 |
| Test to view the monthly salary report | | | | | |
| 20 | Test to check if the salary sheet is shown for date greater than the current date | April 2020 | Error | As Expected | 101 |
| 21 | Test to see if the correct report for the selected date is shown or not | March 2013 | The data from the database is retrieved and correctly shown in the data grid view | As Expected | 101 |
| Test to view the employee payslips | | | | | |
| 22 | Test to see if the correct payslip for the selected employee for month is shown or not | Employee Name: Bishal  Date : March 2013 | The corresponding values from the database are shown in their respective field boxes. | As Expected | 102 |

## Testing

### Testing the login form actions

### Test 1: Ensure empty fields are not accepted for username and password

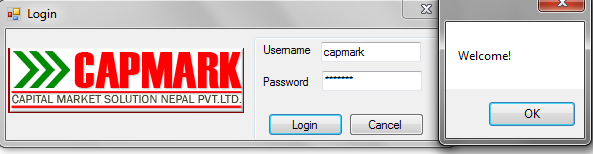


### Test 2: Ensure the system rejects the incorrect login credentials



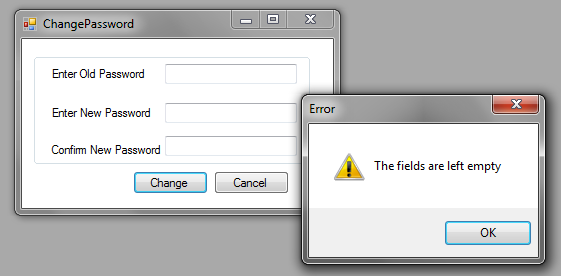
.

### Test 3: Ensure the system accepts the correct login credentials

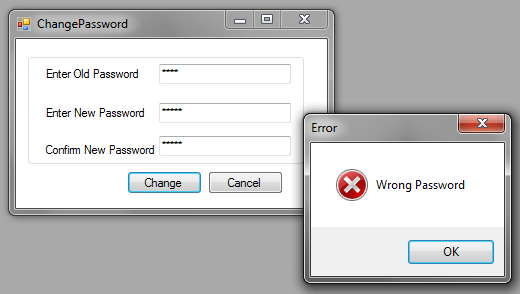


### Testing the change password

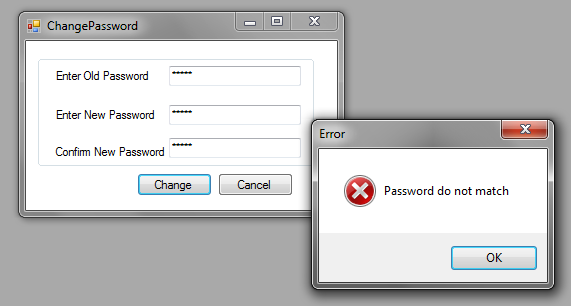
### Test 4: Ensure the empty fields are not accepted



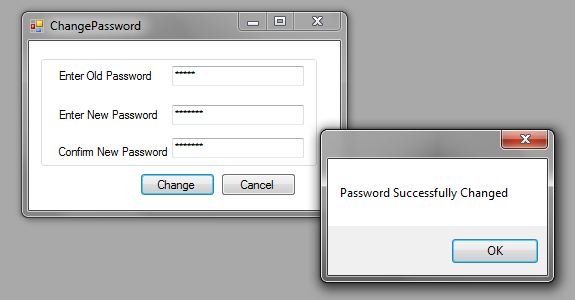
### Test 5: Ensure the wrong password is not accepted



### Test 6: Ensure the different values for new password and confirm passwords are rejected.

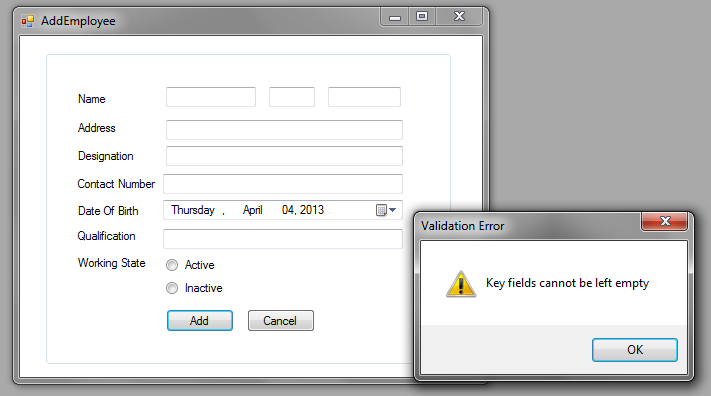


### Test 7: Test to see that validated values are accepted.

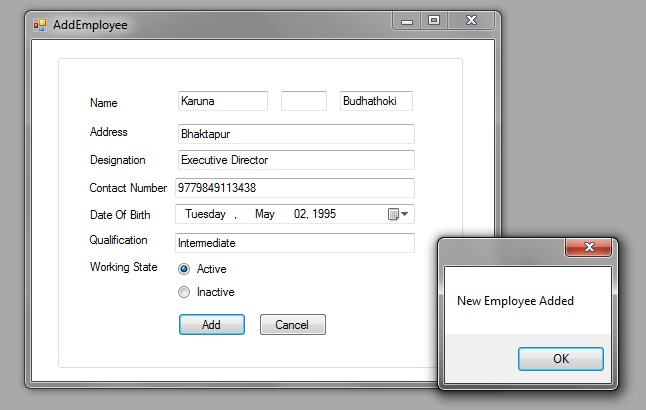


### Testing the action in Add Employee Form

### Test 8: Ensure the empty fields are not accepted

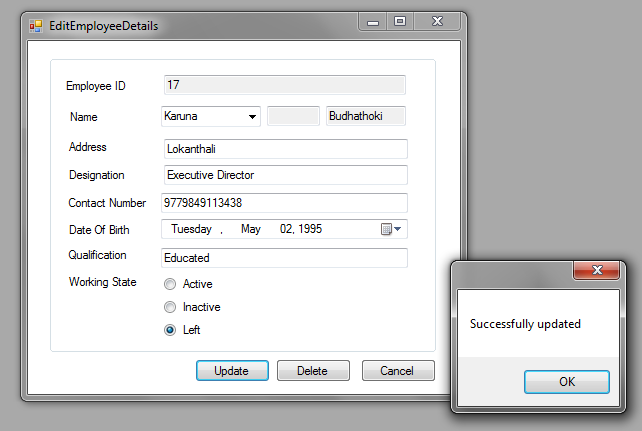


### Test 9: Check to see if the valid values are accepted



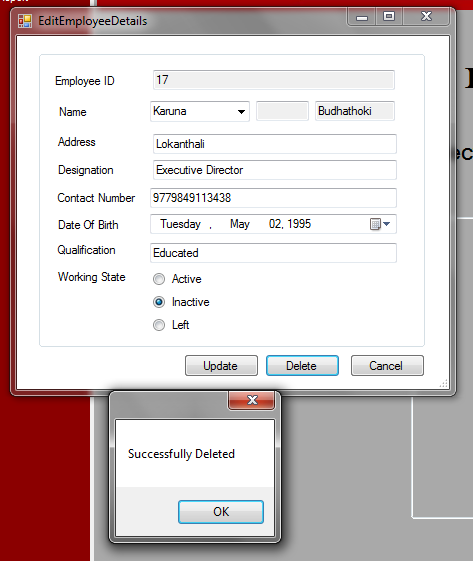
### Testing for Editing the Employee Details

### Test 10: Ensure that the new values are updated



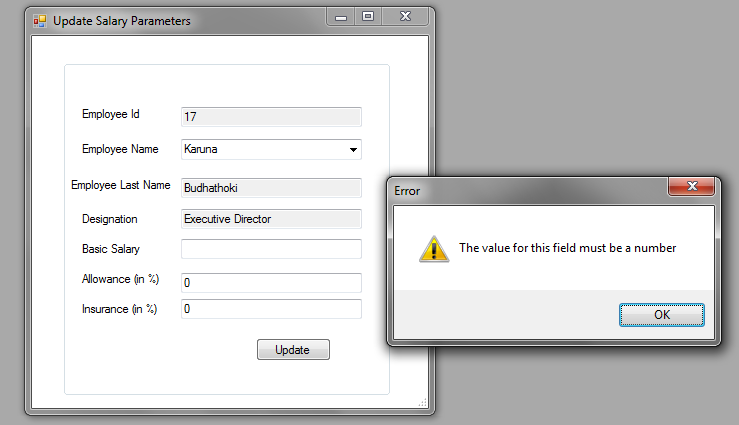
### Testing for deleting the employee record

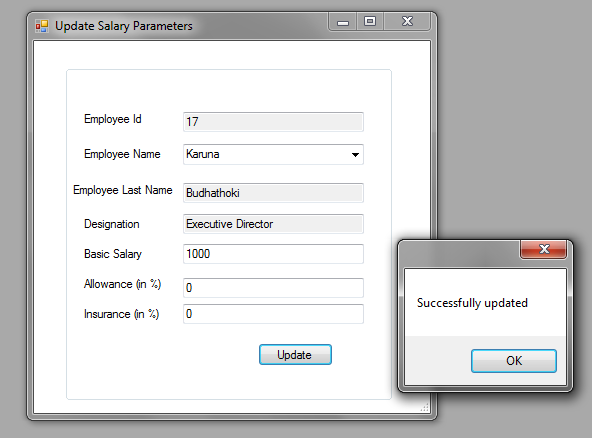
### Test 11: Check if the employee is deleted on clicking delete button



### Testing for assigning of salary parameters

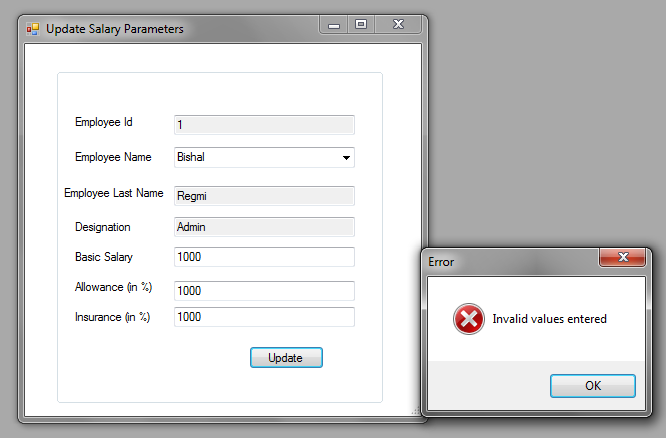
### Test 12: Test to see if Basic Salary field does not accept other character than numbers

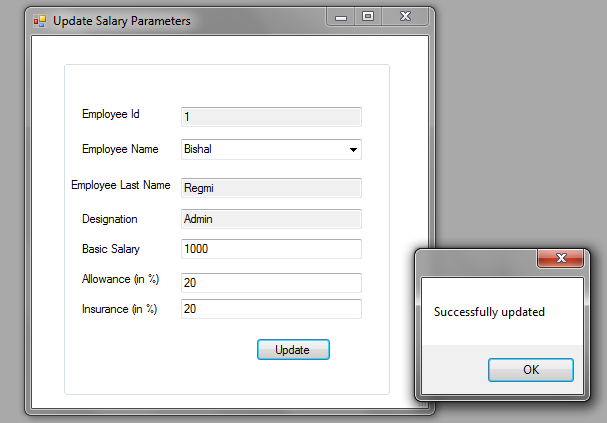




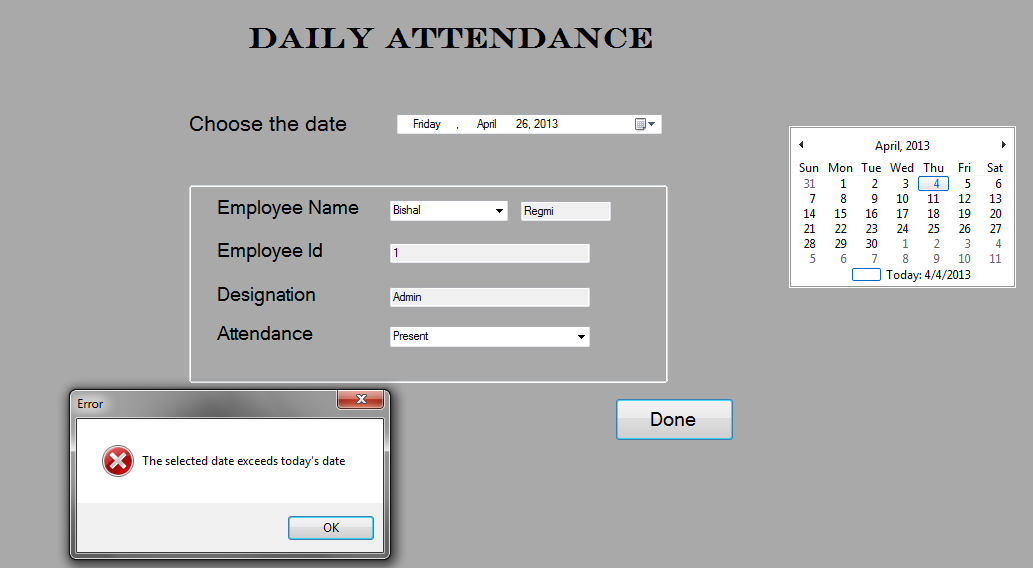
### Test 13: Ensure that the allowance and insurance fields do not allow numeric values more than 100

### Test 14 : Ensure that the values are updated in the database

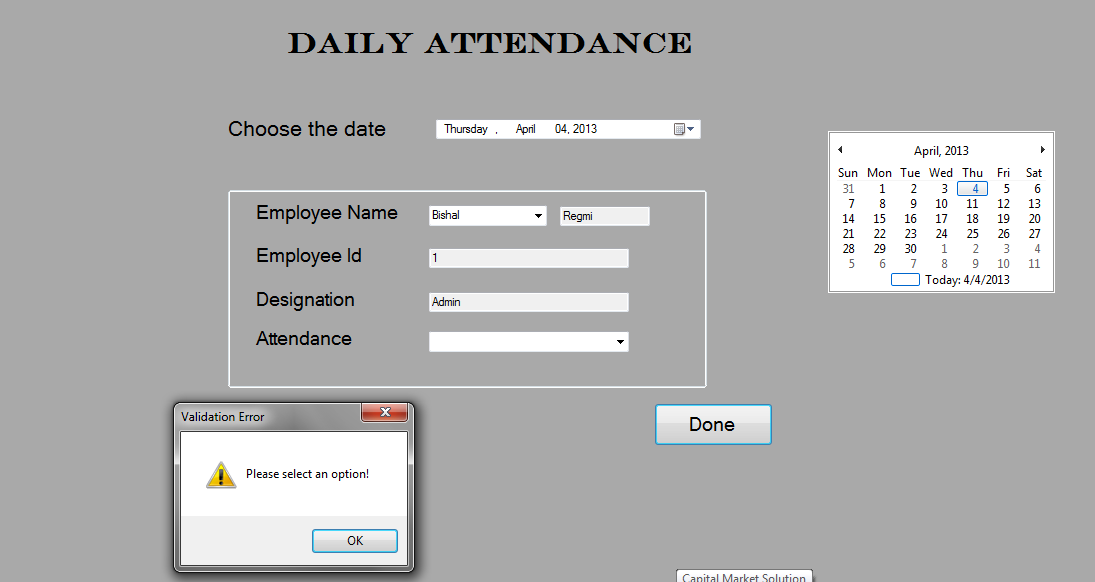




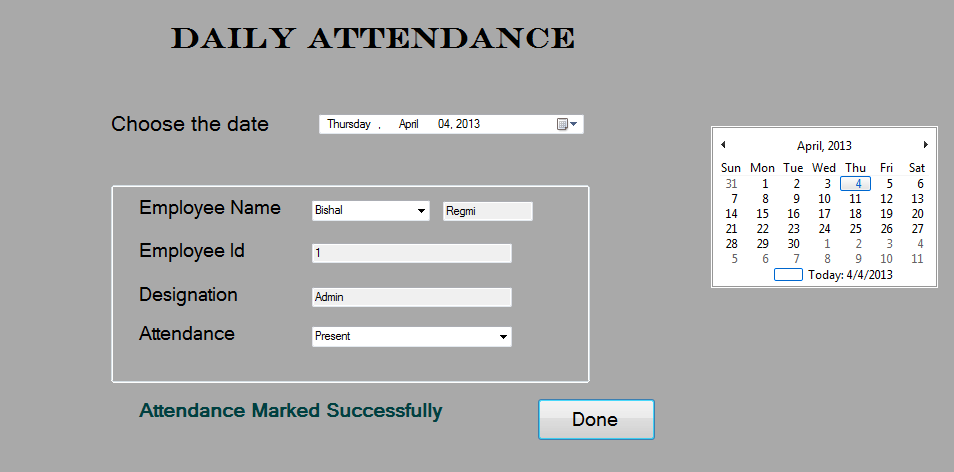
### Test 15: Test to check if the attendance is allowed for date greater than the current date



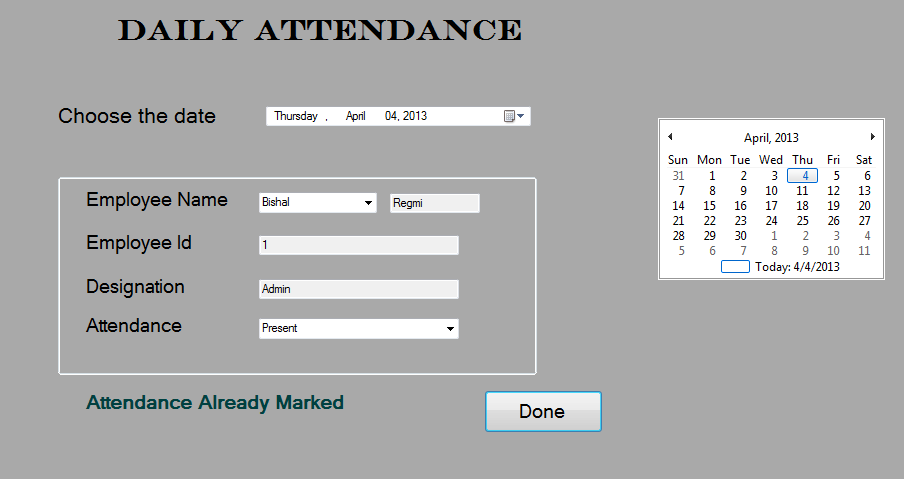
### Test 16: Test to check if the attendance field is allowed empty



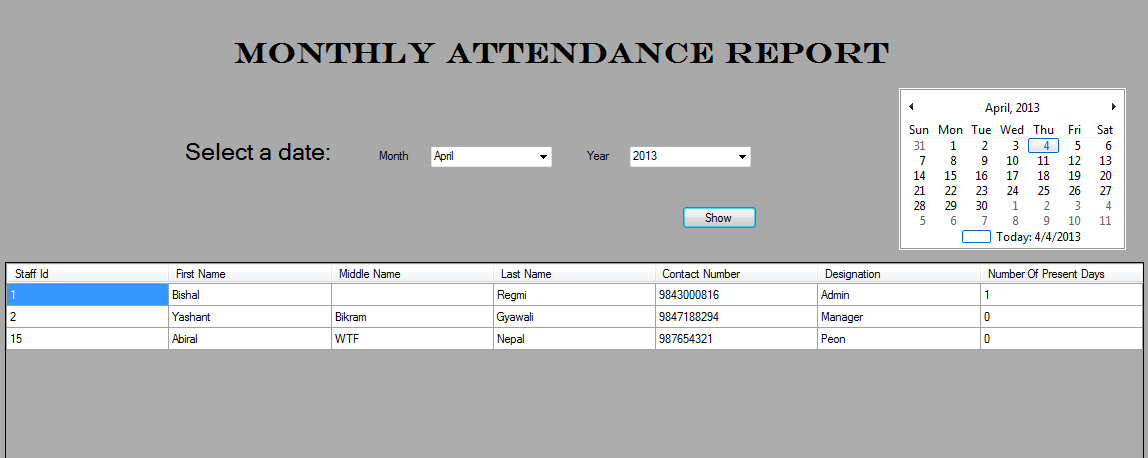
### Test 17: Test to check if the valid data are updated or not



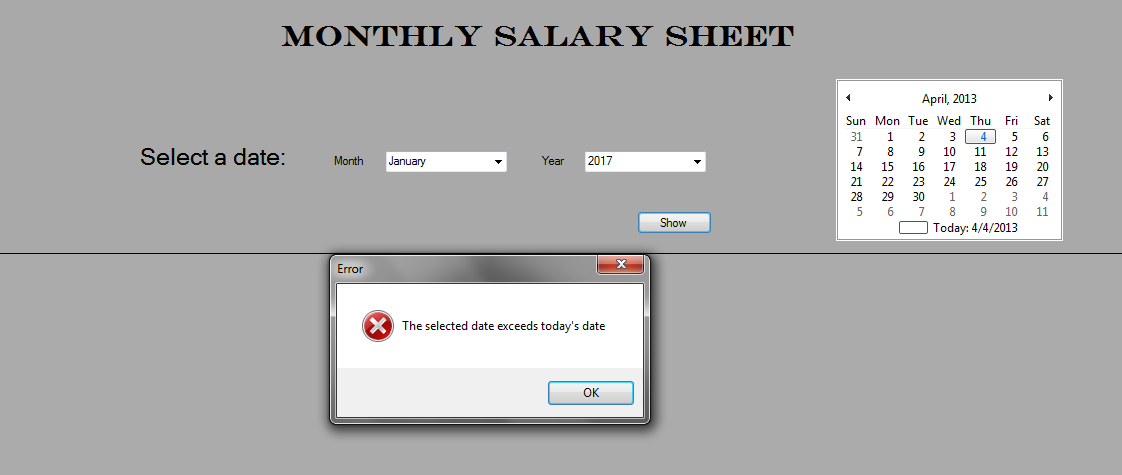
### Test 18: Test to check if the attendance for the same employee for same date is allowed for more than once



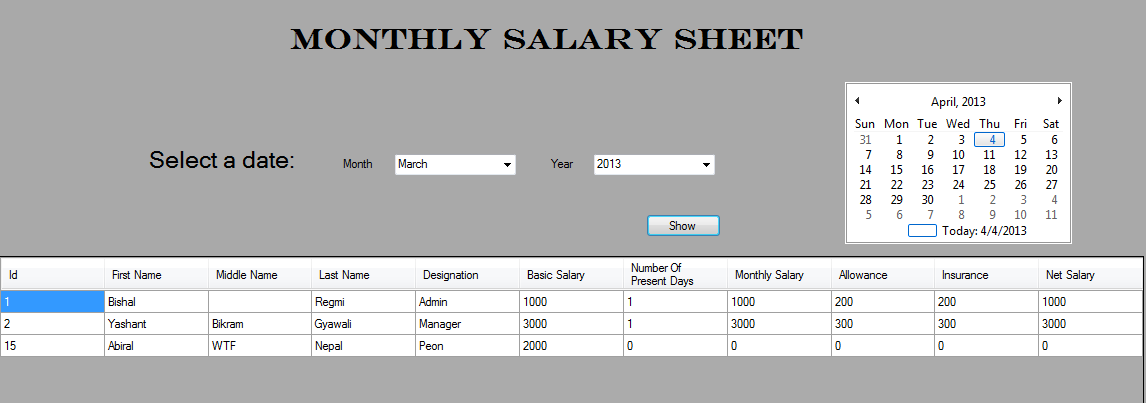
### Test 19: Test to see if the correct report for the selected date is shown or not



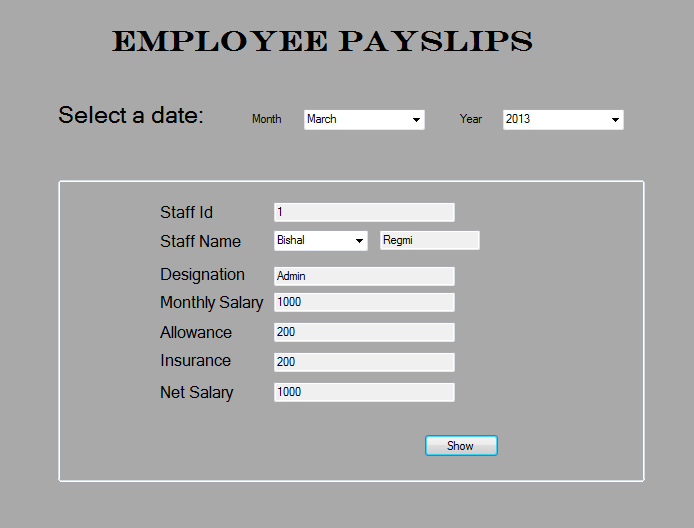
### Test 20: Test to check if the salary sheet is shown for date greater than the current date



### Test 21: Test to see if the correct report for the selected date is shown or not



### Test 22: Test to see if the correct payslip for the selected employee for month is shown or not



# Implementation

## Implementation Plan

After having a meeting with Mr. Paudel, we concluded to adopt the direct changeover technique for the implementation of the system. Since the old system is a manual system and the new system has a lot more advantages over it, other strategies would not be sensible and feasible.

## User Training

The users might need some basic training before operating the new system. The user may find trouble in navigation of the menus and input operations. I will demonstrate the software to each and every users providing them instruction about using the specific features of the system.

When the implementation is made, I will be giving a thorough demonstration to make sure that the user finds the system easy and flexible to use. Also, a training session will be provided to user for the use and maintenance.

Also, users may refer to the user documentation provided. At the same time, I will be visiting the office many times to help the user train.

## User Testing

After the complete development of the system, it was taken into Beta Testing. Mr. Paudel was given to test the system. Following were the tests he made in the system.

1. Login with the correct username and passwords
2. Login with the incorrect username and password
3. Changing the passwords
4. Adding the new employee
5. Editing details about employee
6. Deleting employee details
7. Viewing and updating salary parameters
8. Viewing the employee details
9. Making the attendance updates with valid and invalid sets of data
10. Viewing the monthly attendance report
11. Viewing the monthly salary report
12. Viewing the perspective employee payslips.

All the tests were successful. There were no errors in the system. Mr. Paudel was very happy and thankful for the new system without flaws and greater efficiency.

## User Acceptance

# **3.4. Appropriateness of Structure and Exploitation of Available Resources**

## 3.4.1. Use of Available Resources

### Hardware used for development:

* Pentium IV processor with processor speed of 2.4 GHz
* 3 GB RAM
* 250 GB Hard disk

### Software used for development:

1. Windows 7 was the used operating system.
2. Microsoft Visual Studio 2008 was used as IDE to code the system and create the setup
3. Microsoft SQL Server 2005 was used as Database Management System.
4. Microsoft Word 2007 use to prepare the documentation and reporting
5. Microsoft Visio 2007 used to prepare the diagrammatic explanations like Data Flow Diagrams, System Flowcharts.

## Problem Log

|  |  |  |
| --- | --- | --- |
| Date | Problem Encountered | Action Taken |
| 27th January, 2013 | The connection with the database could not be made | Technician was consulted to correct the problem |
| 12th February, 2013 | IDE crashed due to some technical problems. | I had to re-install Visual Studio 2008 |
| 20th February, 2013 | The database operation could not be made due to the type mismatch of the field property and the database property | Validations and masking methods for the fields were adopted |
| 21st February, 2013 | There were errors in the queries for updating and displaying the salary and attendance reports | Project counselor was consulted for the correction. |
| 25th February 2013 | Run-time errors were shown by the program due to some incomplete coding. | The codes were updated with the help of technician. |
| 26th February, 2013 | There were problems in the tab controls for the user interface. | The form were re-designed |
| 28th February, 2013 | An unexpected power failure caused a total loss of data. | Back-ups were taken regularly for just such a case. The back-ups restored the system to the previous day’s progress. |

5. Evaluation

# 5.1 Degree of Success

The original objectives of the system, their accomplishments and the evidences are enlisted below:

1. **Objective:** To design a fully organized system that works without any or little human intervention.  
   **Accomplishment:** For the calculation of monthly salary for the employees no manual formulae need to execute. The monthly salary is calculated automatically and is shown in the report view. **Evidence:** The monthly salary of the employee needs no user input formula to calculate. It is calculated automatically when the salary sheet is viewed and is stored in the database.
2. **Objective:** To introduce a graphical user interface to make the program user-friendly and convenient to use.  
   **Accomplishment:** The use of forms and the user controls are the examples of Graphical User Interface. **Evidence:** All the forms used in the system are GUI based.
3. **Objective:** To introduce search facility to view the staff details and perspective salary sheets.  
   **Accomplishment:** The user can select view the details, attendance, and salary reports about a particular employee by selecting an option from the combo boxes. **Evidence:** To view the reports, the user selects values from the comboboxes. For example, to view payslip, user selects month, year and employee name and the details are displayed automatically.
4. **Objective:** To avoid redundancy of data.  
   **Accomplishment:** The attendance for the same day for same employee is not allowed to be recorded twice. This avoids redundancy and error in calculating the salary. **Evidence:** Validation checks are made to avoid redundancy. The daily attendance for the same employee for the same date cannot be made more than once.
5. **Objective:** To use Input forms to enter record and output forms or reports to display the salary sheet.  
   **Accomplishment:** Simple forms are used to enter data and grid views are used to display records. **Evidence:** The whole system design is menus and form based. The reports are displayed in data grid views.
6. **Objective:** To minimize errors while record is being entered to the system.   
   **Accomplishment:** Validations for data entered for the respected fields are made. This makes the database values less prone to errors. **Evidence:** Enough validation and verification checks and field maskings are done to minimize errors.

# 5.2 Evaluation of User’s Response

Mr. Paudel was very much thankful and happy to receive the new system. His reviews about the new system were very good. The User Acceptance letter shows that he is very much pleased to see the new system in his office. However, limitations such as inability to print the payslips were mentioned by him. Overall response for the new system was appreciable.

# Desirable Extensions

Almost all the objectives as mentioned earlier have been accomplished. The new system is fast to use, has security and also the search results are produced instantly. Data repetition has been minimized through the use of Primary Keys. The system is also automatic at some extent where required. However, there are few extension that can be further made in the system.

* 1. A printout feature for the payslips of the employees can be added.
  2. Fingerprint scanners can be used to record daily attendance.