**CHAPTER 1**

**INTRODUCTION**

**1.1 PROJECT OVERVIEW**

The project entitled “Transport Management System”is developed by using the Microsoft Visual Studio 2008 as front end and SQL Server as back end.

Itis used to develop the centralized database system for accessing the information by multiple users from different places. It helps to generate reports for all the details of buses, collections, drivers, conductors, schedules, expenses and used to view the average performance of the drivers and conductors. This project helps the transport to get the updated details of the buses easily and frequently. Itgives alerts for schedules like due dates, insurance, bus services, etc. These schedules are followed as per the date registered.

The project consists of four modules namely master entry module, daily collections module, expense module, schedule module and report generation module. Every bus has unique id and helps to know about the details of bus with the collections and expenses. The modules help to generate reports for all the details.

* 1. **OBJECTIVES**

The main objective of the projectis used to develop the centralized database system which maintains bus collections and expenses. So the information can be shared commonly. The administrator should maintain the data stored in the database. The project helps transport to minimize the work of record keeping.

The application mainly helps the administrator to know the details of bus, collections, expenses and schedules. Itgives alerts for schedules like due dates, insurance, bus services, etc. These schedules are followed as per the date registered. So that the late work can be avoided. Reports can be generated for bus, driver, conductor, collection, expense, schedule, etc., The average performance of the driver and conductor can be viewed by the reports. It helps to identify the best driver and conductor among all others.

**CHAPTER 2**

**SYSTEM ANALYSIS**

**2.1 EXISTING SYSTEM**

In existing system, all entries are made in paper like employee, bus, collections, profits and expense details. Transport companies used notebooks to make a note on the collections of buses by making calls to the office and there was a problem in calculating exact profit with expenditure and also maintenance. Reports cannot be generated and alerts cannot be given for schedules. It is difficult to analyse the performance of the driver and conductor.

**2.1.1 Drawbacks**

Following are the drawbacks of the existing system:

* Time consumption
* More paper work is needed
* No security
  1. **PROPOSED SYSTEM**

The proposed system helps administrator where the system is fully computerized and every detail will be stored in a database. Project is to ensure user friendly and more interactive to the administrator. It helps to reduce time by eliminating much manual paper work. It provides centralized database for updating the details of all the buses from anywhere. So the transport competent authorities of the bus companies can effectively manage the real-time data and the related records stored in the database.

Reports can be easily generated for collections, profits, expenses for every buses and performance of the driver can be viewed by making notes on miles, bus conditions, diesel expenses, etc. The system helps to find the annual report with profit in secure manner and consumes less time in generating results.

**2.2.1 Advantages**

Following are the advantages of the proposed system:

* Easy retrieval of data
* More security
* User friendly
* Less paper work
  1. **FEASIBILITY STUDY**

The feasibility study is carried out to test whether the proposed system is worth or not. Having an established system, one has to determine whether an alternative system is feasible compared to existing system. An analysis of the ability to complete a project successfully, it is necessary to take into account legal, economic, technological, scheduling and other factors. Three key considerations involved in feasibility analysis are

* Technical feasibility
* Economical feasibility
* Operational feasibility

**2.3.1 Technical Feasibility**

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. It centres on the existing computer system and to what extent it can support the proposed addition. This involves financial consideration to accommodation technical feasibility. In technical feasibility the following issues are taken into consideration. Whether the required technology is available or not and whether the required resources are available.

SqlServer is a complete business intelligence database platform that provides efficient features and functionality to build both classic and innovative kinds of application.

**2.3.2 Economical Feasibility**

Once the technical feasibility is established, it is important to consider the monetary factors also. Since it might happen that developing a particular system may be technically possible but it may require huge investments and benefits may be less. For evaluating this, economic feasibility of the proposed system is carried out.

The SqlServer license and the C# components are available with Microsoft Visual Studio and then the decision is made to design and implement the system.

**2.3.3 Operational Feasibility**

The purpose of operational feasibility is to determine whether the new system will be used if it is developed and implemented or will there be resistance from the users that will take the possible application benefits. Also it includes the level of acceptance of the system by the user.

The parameters given to the application is less so that any normal user can access the application easily. As the need of this type of system is very high, there will be a great level of acceptance by the user, as time saving is more important for users.

**CHAPTER 3**

**SYSTEM SPECIFICATION**

**3.1 HARDWARE SPECIFICATION**

Processor : Pentium IV

Hard Disk : 200 MB

RAM : 512 MB

**3.2 SOFTWARE SPECIFICATION**

Operating system : Windows XP

Front-end : Visual Studio 2008

Back-end : SQL Server 2005

Programming language : C#

**CHAPTER 4**

**SOFTWARE DESCRIPTION**

**4.1 FRONT END**

**OVERVIEW OF .NET**

.NET is a Software Platform. It is a language-neutral environment for developing rich .NET experiences and building applications that can easily and securely operate within it. When developed applications are deployed, those applications will target .NET and will execute wherever .NET is implemented instead of targeting a particular hardware/OS combination. The components that make up the .NET platform are collectively called the .NET Framework.

The .NET Framework is a managed, type-safe environment for developing and executing applications. The .NET Framework manages all aspects of program execution like allocation of memory for the storage of data and instructions, granting and denying permissions to the application, managing execution of the application and reallocation of memory for resources that are not needed.

The .NET framework is designed for cross-language compatibility. Cross-language compatibility means an application written in Visual Basic .NET may reference a DLL file written in C#. A Visual Basic .NET class might derived from a C# class or vice versa. There are different types of application, such as Windows-based applications and Web-based applications. To make communication on distributed environment to ensure that code be accessed by the .NET Framework can integrate with any other code. C# is an elegant and type-safe object-oriented language that enables developers to build a variety of secure and robust applications that run on the .NET Framework.

Objectives of .net framework

* To provide a consistent object-oriented programming environment whether object codes is stored and executed locally on Internet-distributed, or executed remotely.
* To provide a code-execution environment to minimizes software deployment and guarantees safe execution of code.
* Eliminates the performance problems.

There are different types of application, such as Windows-based applications and Web-based applications. To make communication on distributed environment to ensure that code be accessed by the .NET Framework can integrate with any other code.

Features of .Net Framework

* Language interoperability
* Fully object-oriented languages
* Common runtime engine shared by all languages
* Base class library usable by all languages
* Simplified deployment
* Better security
* Better performance

# **.NET FRAMEWORK ARCHITECTURE**

The .Net framework sits on top of the windows operating system. It consists of the following components:

* Common Language Runtime(CLR)
* Class Libraries
* Common Language Specification(CLS)

**Figure 4.1 .Net Architecture**

# C# programs run on the .NET Framework, an integral component of Windows that includes a virtual execution system called the common language runtime (CLR) and a unified set of class libraries. The CLR is the commercial implementation by Microsoft of the common language infrastructure (CLI), an international standard that is the basis for creating execution and development environments in which languages and libraries work together seamlessly.

**COMMON LANGUAGE RUNTIME (CLR)**

The CLR ids described as the execution engine of .NET. It provides the environment within which the programs run. CLR that manages the execution of programs and provides core services, such as code compilation, memory allocation, thread management and garbage collection.

Through the Common Type System (CTS), it enforces strict type safety and it ensures that the code is executed in a safe environment by enforcing code access security. The software version of .NET is actually the CLR version.

**CLASS LIBRARIES**

Class library is the second major entity of the .Net Framework which is designed to integrate with the common language runtime. This library gives the program access to runtime environment. The class library consists of lots of prewritten code that all the applications created in VB.NET and Visual Studio.NET will use. The code for all the elements like forms, controls and rest in VB.NET applications actually comes from the Class Library.

**COMMON LANGUAGE SPECIFICATION**

If the user wants the code which one can write in a language to be used by programs in other languages then it should adhere to the Common Language Specification (CLS). The CLS describes a set of features that different languages have in common. The CLS defines the minimum standards that .NET language compilers must conform to, and ensures that any source code complied by a .NET compiler can interoperate with the .NET Framework.

**Figure 4.2 Compilation in .Net**

# Source code written in C# is compiled into an intermediate language (IL) that conforms to the CLI specification. The IL code and resources, such as bitmaps and strings, are stored on disk in an executable file called an assembly, typically with an extension of .exe or .dll. When the C# program is executed, the assembly is loaded into the CLR, which might take various actions based on the information in the manifest. Then, if the security requirements are met, the CLR performs just in time (JIT) compilation to convert the IL code to native machine instructions. The CLR also provides other services related to automatic garbage collection, exception handling, and resource management.

**OVERVIEW OF C#**

C# introduces several language extensions including Generics, Anonymous Methods, Iterators, Partial Types and Nullable Types.

Generics permit classes, structs, interfaces, delegates and methods to be parameterized by the types of data they store and manipulate. Generics are useful because they provide stronger compile-time type checking, require fewer explicit conversions between data types and reduce the need for boxing operations and run-time type checks.

Anonymous methods allow code blocks to be written in-line where delegate values are expected. C# supports the creation of closures where anonymous methods access surrounding local variables and parameters.

Iterators are methods that incrementally compute and yield a sequence of values. Iterators make it easy for a type to specify how the foreach statement will iterate over its elements.

Partial types allow classes, structs and interfaces to be broken into multiple pieces stored in different sources files for easier development and maintenance. Additionally, partial types allow separation of machine-generated and user-written parts of types so that it is easier to augment code generated by a tool.

**4.1.1 Features of C#**

Some of the features available in C# as follows

* The C# features provides a null able value types, enumerations, delegates, lambda expressions and direct memory access, which are not found in Java.
* C# supports generic methods and types, which provide increased type safety and performance, and integrators.
* C# struct is like a lightweight class. It is a stack-allocated type that can implement interfaces but does not support inheritance.

**4.2 BACK END**

Microsoft SQL Server is a relational database management system developed by Microsoft Corporation. As a database, it is a software product whose primary function is to store and retrieve data as requested by other software applications, be it those on the same computer or those running on another computer across a network (including the Internet). It is mainly used to fetch the data requested by any other software from the data stored in the server itself. The software, which requests the, data from the server, could be on the same computer or either could be located away on a different computer connected remotely via the internet.

Microsoft SQL 2005 Yukon was launched in October 2005 by the Microsoft Corporation. This version was the successor of the Microsoft SQL 2000. Microsoft SQL is designed to help with the workload of a wide range of audiences from small application based organization to large and vast data processing organizations which SQL Server 2005 is the integration of a .NET compliant language such as C#, ASP.NET or VB.NET to build objects (stored procedures, triggers, functions, etc).

SQL Server 2005 introduced DMVs (Dynamic Management Views), which are specialized views and functions that return server state information that can be used to monitor the health of a server instance, diagnose problems, and tune performance. Microsoft SQL Server 2005 also allows user-defined composite types (UDTs) to be defined and used. It also makes server statistics available as virtual tables and views (called Dynamic Management Views or DMVs). In addition to tables, a database can also contain other objects including [views](http://en.wikipedia.org/wiki/View_(database)), [stored procedures](http://en.wikipedia.org/wiki/Stored_procedure), [indexes](http://en.wikipedia.org/wiki/Index_(database)) and [constraints](http://en.wikipedia.org/wiki/Constraint_(database)), along with a transaction log.

**4.2.1 Features**

Some of the features available in Microsoft SQL SERVER 2005 as follows

* Support for the XML data management and relational data management. For error handling, the Microsoft SQL 2005 was supported with the try-catch procedure.
* Query processor handles concurrent execution of queries in a more efficient way.

**CHAPTER 5**

**PROJECT DESCRIPTION**

**5.1 MODULE DESCRIPTION**

The application consists of following modules

* + - Master entry
    - Daily collections
    - Expenses
    - Schedules
    - Report generation

**5.1.1** **Master entry**

Thismodule contains information about the bus, employee, salary details, diesel prices and schedule types**.** Administrator will register all the details of the buses, employees, schedules, diesel prices, salary of employee, etc.., Bus detail contains bus number, engine number, chassis number and date of purchase. Employee detail contains drivers and conductors personal information along with their images. Diesel price and salary are updated whenever the changes required. Schedule types are entered with schedule names along with specific colours. All these details are stored in the database and used for the future use.

**5.1.2 Daily collections**

The daily collections of all the buses are entered by the particular employees on every day by logging into the application giving username and password. The image of the driver and conductor on that particular day was displayed along with their names. The

unique id of every bus will be selected for the particular date. It also contains diesel expenses and other expenses for every bus to calculate the total expenses for buses. The total profit can be calculated from the collections and expenses for each bus.

**5.1.3 Expenses**

This module contains information about the expense details entered by both administrator and user. Expenses of every bus are calculated for the expense type entered with the amount. The unique id for every bus will be selected. Expense type will be services like brake, tyre, seat changes and other expenses are toll fee, bank due, etc., So that monthly expense for every bus can be maintained easily.

**5.1.4 Schedules**

This module contains information about schedules entered in schedule type. Schedules can be entered for bus services, bank dues, insurance, etc., So that the alerts can be made for the schedules correctly. Schedule types are entered with specific colours and schedules for each bus are entered with reminder date and old date by selecting the unique id of the bus. Alert messages for every schedule will be displayed with different colours specified in schedule entry form.

**5.1.5 Report generation**

This module contains reports for the stored details as daily, weekly, monthly and yearly. Reports are generated for the buses, collections, expenses, schedules and average report for drivers and conductors. Reports will display the final total for the transport details of the selected dates.

Reports help to view the total profit and expense of the transport and to maintain the details properly. The average performance of the drivers and conductors for particular buses are generated in report and it helps to view the average performance of each driver and conductor to know about the employees work.

**5.2 DATAFLOW DIAGRAM**

User

Administrator

**Figure 5.1 DFD-Level 0**

User

Administrator

Expense table Schedule table Employee table Bus table Collection table

**Figure 5.2 DFD-Level 1**

**5.3 E-R DIAGRAM**

COLLECTION

PROFIT

M M

Collector

Has

1 1

BUS

1 1

Bus\_schedule

Has

M

M

SCHEDULE

EXPENSE

**Figure 5.3 Entity Relationship diagram**

**5.4 DATABASE DESIGN**

The database design involves the creation of tables. Tables are represented in physical database as stored files. They have their own independent existence. A table consists of rows and columns. Each column corresponds to a piece of information called field. A set of fields constitutes a record. The record contains all the information specific to a particular member.

**Table 5.1 Bus details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Description** | **Constraints** |
| Bus id | Nvarchar(7) | ID for the bus | Primary key |
| Reg no | Nvarchar(10) | Register number for the bus | Not null |
| Owner | Nvarchar(20) | Owner of the bus | Not null |
| Engine no | Int(7) | Particular engine number for the bus | Not null |
| Chassis no | Nvarchar(10) | Particular chassis number | Not null |
| DOP | date | Date in which bus purchased | Not null |

**Table 5.2 Employee details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Description** | **Constraints** |
| Eid | Int(5) | Employee Id | Primary key |
| Empname | Nvarchar(20) | Employee names | Not null |
| Address | Nvarchar(50) | Address of the employee | Not null |
| City | Nvarchar(10) | City of the employee | Not null |
| Mobile | Int(10) | Unique constraint | Not null |
| Age | Int(2) | Age of the employee | Not null |
| Dlno | Nvarchar(15) | License number of the employee | Not null |
| Department | Nvarchar(10) | Designation | Not null |
| Image | Image | Image of the employees | Not null |

**Table 5.3 Collection details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Description** | **Constraints** |
| Cid | Int(5) | Collection Id | Primary key |
| Cdate | date | Collection for particular date | Not null |
| Driver | Nvarchar(20) | Driver of the bus | Not null |
| Conductor | Nvarchar(20) | Conductor of the bus | Not null |
| Bus id | Nvarchar(7) | Bus Id | Foreign key |
| Collection | decimal | Daily collection of the bus | Not null |

**Table 5.4 Profit details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Description** | **Constraints** |
| Bid | Int(5) | Bus Id | Foreign key |
| Dlit | Decimal(10) | Diesel in litres | Not null |
| Damount | Decimal(10) | Diesel amount | Not null |
| Salary | Decimal(10) | Daily amount for employees | Not null |
| Unionexp | Decimal(10) | Amount for the employee unions | Not null |
| Cleaner | Decimal(10) | Salary for cleaners | Not null |
| Ticket return | Decimal(10) | Amount of ticket returned | Not null |
| Agent | Decimal(10) | Agents amount | Not null |
| Celebration | Decimal(10) | Amounts for celebrations | Not null |
| Extra | Decimal(10) | Other extra expenses | Not null |
| Workshop | Decimal(10) | Service expenses | Not null |
| Connection | Decimal(10) | Connection amount | Not null |
| Net total | Decimal(10) | Total profit of the bus | Not null |

**Table 5.5 Expense details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Description** | **Constraints** |
| Expid | Int(5) | Expense Id | Primary key |
| EDate | date | Expenses in particular date | Not null |
| Bus id | Nvarchar(7) | Bus Id | Foreign key |
| Exptype | Nvarchar(10) | Expense type for bus | Not null |
| Amount | Decimal(10) | Total amount for expense | Not null |
| Remark | Nvarchar(20) | Remarks for the expenses | Not null |

**Table 5.6 Schedule details**

|  |  |  |  |
| --- | --- | --- | --- |
| **Field name** | **Data type** | **Description** | **Constraints** |
| Sid | Int(5) | Schedule Id | Primary key |
| Bus id | Nvarchar(7) | Bus Id | Foreign key |
| Type | Nvarchar(10) | Schedule type for the bus | Not null |
| Day | Int(5) | Day for the schedule | Not null |
| Month | Int(5) | Month for the schedule | Not null |
| Year | Int(5) | Year of the schedule | Not null |
| Old date | date | Old date in which services done | Not null |
| Last date | date | Last date for the schedule | Not null |
| Reminder date | date | Last date to remind | Not null |

**5.5 INPUT DESIGN**

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

In this system, the user can login into the application by giving the user name and password. Then they can enter employee details, bus details, collection details and expense details. The users are allowed to enter the needed details to generate reports.

The objectives to guide the design of the input process focus on

* Effectiveness in reports
* Accuracy of data
* Easy to use
* Consistency
* Simplicity
* Attractiveness

Consistency means the screens should group the data of similar nature together. Simplicity refers to keeping the screen simple. Attractiveness should be of appealing design, which should please the user.

**5.6 OUTPUT DESIGN**

Computer output is the most important and the direct source of information to the user. Efficient and intelligible output design will improve the system relationships with the user and help in decision. Output design generated refers to the results generated by the system. Output is the key tool to evaluate the performance of software, so the designing of output will be down with great care. It should be able to satisfy the users requirements.

The output of a system can take many forms scheduling details and reports like bus, driver, conductor, collection, scheduling and expense. Finally, the report is viewed by the administrator to know the performance of the driver and conductor.

**CHAPTER 6**

**SYSTEM TESTING**

Any project is said to be successful and reliability only if it is free from errors. Testing the product in all dimensions can ensure it. A thorough testing alone can set the product to high standards. Hence testing is a very crucial phase during the development of the software.

Any software has to be tested with pre-planned strategies. Testing is a process of executing program with the intention of finding errors. A good test case is one that has a high probability of finding an undiscovered error.

**6.1 UNIT TESTING**

Unit testing is the test performed on individual modules. In unit testing, each and every form of transport is tested for correct performance. Every module is tested individually to debug the errors on various operations such as saving, updating, deleting, searching for the data which also includes alphabetical search.

**6.2 INTEGRATION TESTING**

Integration testing indulges in the individual software modules that are combined and tested as a group. The integration testing dictates the order by which modules are available and thus exerts a strong influence on the order in which the modules are written, debugged and unit tested. Report generation will be checked with compatibility of master entry, collections, expenses and schedule modules in order to find whether it is compatible or not.

**6.3 ACCEPTANCE TESTING**

Acceptance testing is done to verify if the application conforms to the requirements defined by the user and also to check whether the report is generated successfully or not. The administrator checks the driver and conductor reports for their average performance.

**6.4 TEST CASES**

A test case is a set of conditions or variables under which a tester will determine if a requirement is partially or fully satisfied.

**Table 6.1 Test case for employee details**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Test case description** | **Expected output** | **Actual result** |
| 1 | Click the Master menu and select Employee | If the employee name “Ram”, address “1/176, Nehru street”, city “Namakkal”, mobile “9360725688”, license “TN30 34682445349” then valid.  If the employee name “Ram3445”, address “1/176, Nehru street”, city “Namakkal”, mobile “9360725688er”, license “TN30\*- 34682445349” then invalid. | PASS  FAIL |
| 2 | Click the save button | If employee details successfully saved and text displayed in message box.  If message appears to fill the respective data in the blank fields. | PASS  FAIL |

**Table 6.2 Test case for bus details**

|  |  |  |  |
| --- | --- | --- | --- |
| **S.No.** | **Test case description** | **Expected output** | **Actual result** |
| 1 | Click the Master menu and select Bus | If the bus id “AN 3099”, reg. number “TN30 AN 3099”, owner “Ramu”, engine number “283776”, chassis number “4857746” date of purchase “15-02-2014” then valid.  If the bus id “AN 3099”, reg. number “TN30 AN 3099”, owner “Ramu467”, engine number “283776nb”, chassis number “4857746op” date of purchase “15-02-2014jii” then invalid. | PASS  FAIL |
| 2 | Click the save button | If bus details successfully saved and text displayed in message box.  If message appears to fill the respective data in the blank fields. | PASS  FAIL |

**CHAPTER 7**

**SYSTEM IMPLEMENTATION**

System implementation is the most important stage of the project when the input design is tuned into practical system. Implementation is the process of converting a new system design into operation. The planning is the first task in the system implementation. Each module is tested individually at the time of development using the sample data and has verified that these modules link together in the way specified in the program specification. First analyzing the problem of this project. At the time of implementation, apply the new concept. The administrator involves to understanding overall system and in being able to carry out effectively the specified task.

In this project, the application is implemented in such a way that the employees have to enter the daily collection details for every buses and the administrator will generate the report for the bus collections. The application file is generated and runs in 3.5 .Net framework. The application had been implemented on users environment and it was successfully executed.

**CHAPTER 8**

**CONCLUSION AND FUTURE ENHANCEMENTS**

**8.1 CONCLUSION**

The project entitled “Transport Management System” is successfully designed and implemented. The main objective of this application is to access the information from different places by using the database as centralized. It is assured that this system facilitates the activities of schedule alerts and reports are generated. It provides the clear knowledge about the process of the transport and their total worth. By this system all the calculations and information can be accessed easily.

This application provides a user friendly approach towards the system. The system has been well developed and is found to satisfy all of the requirements. This hopes that the system will be utilized to its maximum and will do a good job in the long run.

* 1. **FUTURE ENHANCEMENTS**

The system can be further enhanced in future by including the following features

* Updating of diesel price and salary of the employees automatically to do the calculation of the expenses.
* Implement the application as mobile application for the employees to update the current status of the bus.

The new system is designed such that those enhancements can be integrated with current modules easily with less integration work.

**APPENDIX 1**

**A1. SOURCE CODE**

**ScheduleReportForm.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlClient;

using BUS.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

namespace BUS

{

public partial class ScheduleReportForm : Form

{

public ScheduleReportForm()

{

InitializeComponent();

}

private void ScheduleReportForm\_Load(object sender, EventArgs e)

{

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

conn.Open();

string qry1 = "SELECT '\_\_All\_\_' AS Name UNION Select Name

FROM BusMaster ORDER BY Name";

DataSet ds = new DataSet();

SqlDataAdapter da = new SqlDataAdapter(qry1, conn);

da.Fill(ds);

cmbBusName.DataSource = ds.Tables[0].DefaultView;

cmbBusName.DisplayMember = "Name";

conn.Close();

}

private void cmbBusName\_SelectedIndexChanged(object sender,

EventArgs e)

{

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

int busid = General.GetBusID(cmbBusName.Text, General.BUS);

conn.Open();

string qry1 = "";

if (cmbBusName.Text == "\_\_All\_\_")

{

qry1 = "SELECT LastDate,ReminderDate,ScheduleType+'-'+Name+'-'+RegNo

AS Name,Color FROM ScheduleMaster S INNER JOIN BusMaster B ON

S.BusMaster\_ID = B.ID INNER JOIN ScheduleType ST On ST.ID =

S.Type\_ID";

}

else

{

qry1 = "SELECT LastDate,ReminderDate,ScheduleType+'-'+Name+'-'+RegNo

AS Name,Color FROM ScheduleMaster S INNER JOIN BusMaster B ON

S.BusMaster\_ID = B.ID INNER JOIN ScheduleType ST On ST.ID = S.Type\_ID

Where BusMaster\_ID = " + busid;

}

DataSet ds = new DataSet();

SqlDataAdapter da = new SqlDataAdapter(qry1, conn);

da.Fill(ds);

conn.Close();

int length = ds.Tables[0].Rows.Count;

calendar1.CalendarEvents.Clear();

for (int i = 0; i < length; i++)

{

DateTime lastdate =

Convert.ToDateTime(ds.Tables[0].Rows[i]["LastDate"].ToString());

DateTime reminderDate =

Convert.ToDateTime(ds.Tables[0].Rows[i]["ReminderDate"].ToString());

while (lastdate >= reminderDate)

{

Appointment event1 = new Appointment();

event1.StartTime = reminderDate;

event1.EndTime = reminderDate;

event1.BackColor =

System.Drawing.ColorTranslator.FromHtml(ds.Tables[0].Rows[i]["Color"].ToString());

event1.Text = ds.Tables[0].Rows[i]["Name"].ToString();

calendar1.CalendarEvents.Add(event1);

reminderDate = reminderDate.AddDays(1);

dateTimePicker1.Text = DateTime.Today.ToShortDateString();

calendar1.CurrentDate = DateTime.Today;

}

}

}

private void dateTimePicker1\_ValueChanged(object sender, EventArgs e)

{

this.calendar1.CurrentDate = dateTimePicker1.Value;

}

private void calendar1\_CellClick(object sender, DataGridViewCellEventArgs e)

{

if (calendar1.SelectedAppointment != null)

{

string[] value = calendar1.SelectedAppointment.Text.Split('-');

string type = value[0];

string name = value[1];

DialogResult dr = MessageBox.Show("Do you Want to Re-Schedule it?", "Re-

Schedule", MessageBoxButtons.YesNo);

if (dr == DialogResult.Yes)

{

int busid = General.GetBusID(name, General.BUS);

int typeid = General.GetBusID(type, General.SCHEDULETYPE);

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

conn.Open();

string qry1 = "SELECT \* FROM ScheduleMaster Where BusMaster\_ID = " +

busid + " and type\_id =" + typeid;

DataSet ds = new DataSet();

SqlDataAdapter da = new SqlDataAdapter(qry1, conn);

da.Fill(ds);

conn.Close();

DateTime olddate =

Convert.ToDateTime(ds.Tables[0].Rows[0]["OldDate"].ToString());

DateTime lastdate =

Convert.ToDateTime(ds.Tables[0].Rows[0]["LastDate"].ToString());

DateTime reminderDate =

Convert.ToDateTime(ds.Tables[0].Rows[0]["ReminderDate"].ToString());

int day = Convert.ToInt32(ds.Tables[0].Rows[0]["Day"].ToString());

int Month = Convert.ToInt32(ds.Tables[0].Rows[0]["Month"].ToString());

int Year = Convert.ToInt32(ds.Tables[0].Rows[0]["Year"].ToString());

string Format = ds.Tables[0].Rows[0]["Format"].ToString();

int remidnerday =

Convert.ToInt32(ds.Tables[0].Rows[0]["ReminderDay"].ToString());

olddate = lastdate;

if (Format == General.DAY)

{

lastdate = lastdate.AddDays(day);

}

else if (Format == General.MONTH)

{

lastdate = lastdate.AddMonths(Month);

}

else if (Format == General.YEAR)

{

lastdate = lastdate.AddYears(Year);

}

reminderDate = lastdate.AddDays(-1 \* remidnerday);

string qry = "UPDATE ScheduleMaster SET ";

qry += " OldDate = '" + olddate.ToString("yyyy-MM-dd") + "',";

qry += " LastDate = '" + lastdate.ToString("yyyy-MM-dd") + "',";

qry += " ReminderDate = '" + reminderDate.ToString("yyyy-MM-dd") + "'";

qry += " WHERE BusMaster\_ID =" + busid;

qry += " AND Type\_ID=" + typeid;

conn.Open();

SqlCommand cmd = new SqlCommand(qry, conn);

cmd.ExecuteNonQuery();

conn.Close();

dateTimePicker1.Text = DateTime.Today.ToShortDateString();

calendar1.CurrentDate = DateTime.Today;

button1\_Click(sender, e);

MessageBox.Show("Re-Scheduled Successfully");

cmbBusName\_SelectedIndexChanged(sender, e);

}

else if (dr == DialogResult.No)

{

MessageBox.Show("Waiting for Rescheduled");

}

else

{

MessageBox.Show("else");

}

calendar1.SelectedAppointment = null;

}

}

}

}

**MainReport.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.Threading;

using System.IO;

namespace BUS

{

public partial class frmMainReport : Form

{

public frmMainReport()

{

InitializeComponent();

}

private void frmMainReport\_Load(object sender, EventArgs e)

{

dtpfromdate.Value = DateTime.Now;

dtptodate.Value = DateTime.Now;

cbbusname.Items.Clear();

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

conn.Open();

string qry = "SELECT '\_\_All\_\_' AS Name UNION Select Name FROM BusMaster

ORDER BY Name";

SqlDataAdapter da = new SqlDataAdapter(qry, conn);

DataSet ds = new DataSet();

da.Fill(ds);

conn.Close();

int rowcount = ds.Tables[0].Rows.Count;

if (rowcount == 0)

{

MessageBox.Show("No Driver Record Found");

}

else

{

for (int count = 0; count < rowcount; count++)

{

cbbusname.Items.Insert(count,

Convert.ToString(ds.Tables[0].Rows[count]["Name"]).Trim());

}

cbbusname.SelectedIndex = 0;

}

DriverReport dr = new DriverReport();

}

private void btcreate\_Click(object sender, EventArgs e)

{

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

conn.Open();

string qry = "";

if (cbbusname.Text == "\_\_All\_\_")

{

qry = "select \* FROM CollectionMaster WHERE Date between'" +

dtpfromdate.Value.ToString("yyyy-MM-dd") + "' AND '" +

dtptodate.Value.ToString("yyyy-MM-dd") + "' order by Date";

}

else

{

qry = "select \* FROM CollectionMaster WHERE Date between'" +

dtpfromdate.Value.ToString("yyyy-MM-dd") + "' AND '" +

dtptodate.Value.ToString("yyyy-MM-dd") + "' AND BusName='" +

cbbusname.SelectedItem.ToString() + "' order by Date";

}

SqlDataAdapter da = new SqlDataAdapter(qry, conn);

DataSet ds = new DataSet();

da.Fill(ds);

conn.Close();

ReportDocument cryRptDoc = new ReportDocument();

string reportpath =

Path.GetDirectoryName(Application.ExecutablePath).ToString() +

General.GetReportPath(General.MAINREPORT);

cryRptDoc.Load(reportpath);

cryRptDoc.SetDataSource(ds.Tables[0].DefaultView);

crvcollection.ReportSource = cryRptDoc;

cryRptDoc.SetParameterValue("FromDate", dtpfromdate.Text);

cryRptDoc.SetParameterValue("ToDate", dtptodate.Text);

cryRptDoc.SetParameterValue("BusName", cbbusname.SelectedItem.ToString()); }

}

}

**CollectionReport.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.Threading;

using System.IO;

namespace BUS

{

public partial class frmShortCollectionReport : Form

{

public frmShortCollectionReport()

{

InitializeComponent();

}

private void CollectionReport\_Load(object sender, EventArgs e)

{

dtpfromdate.Value = DateTime.Now;

dtptodate.Value = DateTime.Now;

cbbusname.Items.Clear();

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

conn.Open();

string qry = "SELECT '\_\_All\_\_' AS Name UNION Select Name FROM BusMaster

ORDER BY Name";

SqlDataAdapter da = new SqlDataAdapter(qry, conn);

DataSet ds = new DataSet();

da.Fill(ds);

conn.Close();

int rowcount = ds.Tables[0].Rows.Count;

if (rowcount == 0)

{

MessageBox.Show("No Driver Record Found");

}

else

{

for (int count = 0; count < rowcount; count++)

{

cbbusname.Items.Insert(count,

Convert.ToString(ds.Tables[0].Rows[count]["Name"]).Trim());

}

cbbusname.SelectedIndex = 0;

}

DriverReport dr = new DriverReport();

}

private void btcreate\_Click(object sender, EventArgs e)

{

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

conn.Open();

string qry = "";

if (cbbusname.Text == "\_\_All\_\_")

{

qry = "select \* FROM CollectionMaster WHERE Date between'" +

dtpfromdate.Value.ToString("yyyy-MM-dd") + "' AND '" +

dtptodate.Value.ToString("yyyy-MM-dd") + "' order by Date";

}

else

{

qry = "select \* FROM CollectionMaster WHERE Date between'" +

dtpfromdate.Value.ToString("yyyy-MM-dd") + "' AND '" +

dtptodate.Value.ToString("yyyy-MM-dd") + "' AND BusName='" +

cbbusname.SelectedItem.ToString() + "' order by Date";

}

SqlDataAdapter da = new SqlDataAdapter(qry, conn);

DataSet ds = new DataSet();

da.Fill(ds);

conn.Close();

ReportDocument cryRptDoc = new ReportDocument();

string reportpath =

Path.GetDirectoryName(Application.ExecutablePath).ToString() +

General.GetReportPath(General.COLLECTION);

cryRptDoc.Load(reportpath);

cryRptDoc.SetDataSource(ds.Tables[0].DefaultView);

crvcollection.ReportSource = cryRptDoc;

cryRptDoc.SetParameterValue("FromDate", dtpfromdate.Text);

cryRptDoc.SetParameterValue("ToDate", dtptodate.Text);

cryRptDoc.SetParameterValue("BusName", cbbusname.SelectedItem.ToString());

}

}

}

**MonthlyReport.cs**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using CrystalDecisions.CrystalReports.Engine;

using CrystalDecisions.Shared;

using System.Data.SqlClient;

using System.Data.SqlTypes;

using System.Threading;

using System.IO;

namespace BUS

{

public partial class frmMonthlyReport : Form

{

public frmMonthlyReport()

{

InitializeComponent();

}

private void btcreate\_Click(object sender, EventArgs e)

{

string connstring = General.CONNECTION\_STRING;

SqlConnection conn = new SqlConnection(connstring);

conn.Open();

string qry = "";

qry = "up\_GetMonthlyReport @StartDate='" + dtpfromdate.Value.ToString("yyyy-

MM-dd") + "',@endDate='" + dtptodate.Value.ToString("yyyy-MM-dd") + "'";

SqlCommand cmd = new SqlCommand();

cmd.Connection = conn;

cmd.CommandText = qry;

cmd.ExecuteNonQuery();

qry = "SELECT \* FROM MonthlyReport";

SqlDataAdapter da = new SqlDataAdapter(qry, conn);

DataSet ds = new DataSet();

da.Fill(ds);

conn.Close();

ReportDocument cryRptDoc = new ReportDocument();

string reportpath =

Path.GetDirectoryName(Application.ExecutablePath).ToString() +

General.GetReportPath(General.MONTHLYREPORT);

cryRptDoc.Load(reportpath);

cryRptDoc.SetDataSource(ds.Tables[0].DefaultView);

crvcollection.ReportSource = cryRptDoc;

cryRptDoc.SetParameterValue("FromDate", dtpfromdate.Text);

cryRptDoc.SetParameterValue("ToDate", dtptodate.Text);

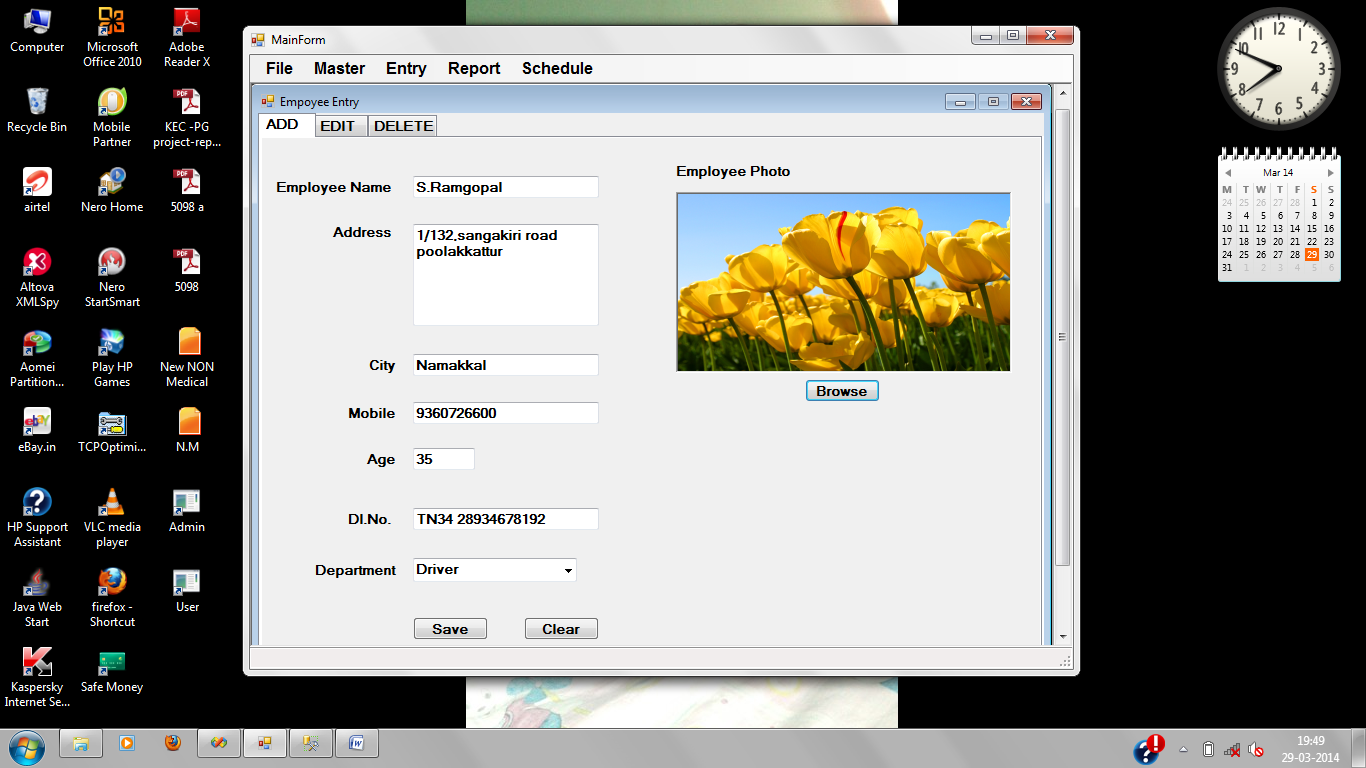
}

}

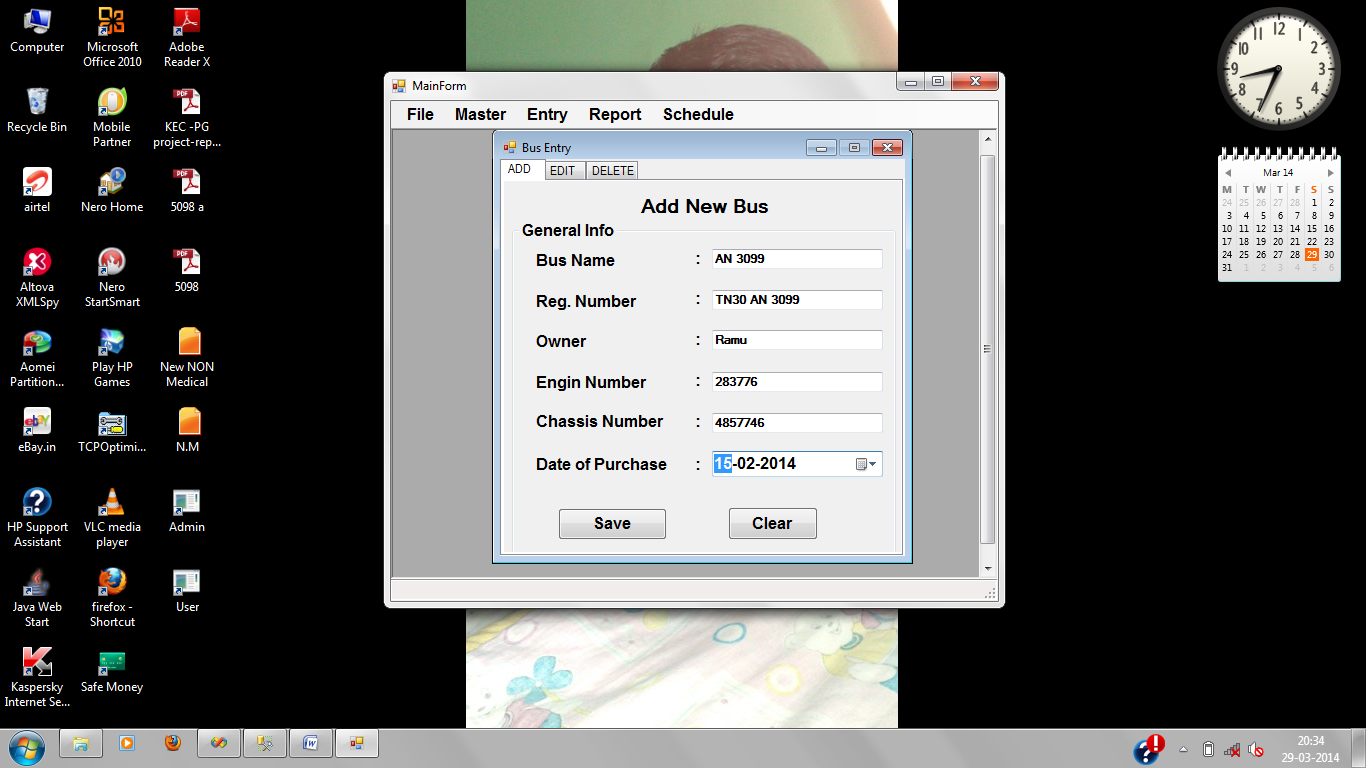
}

**APPENDIX 2**

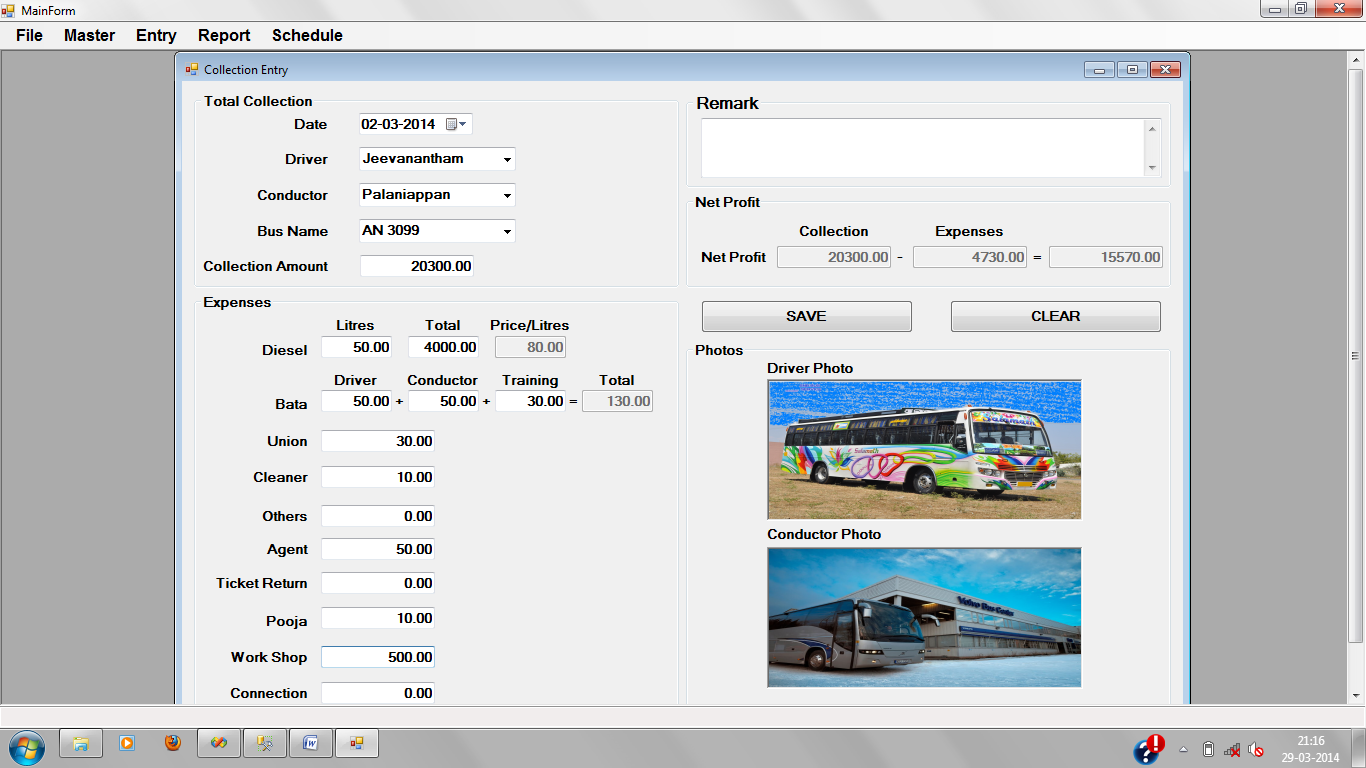
**A2. SCREEN SHOTS**



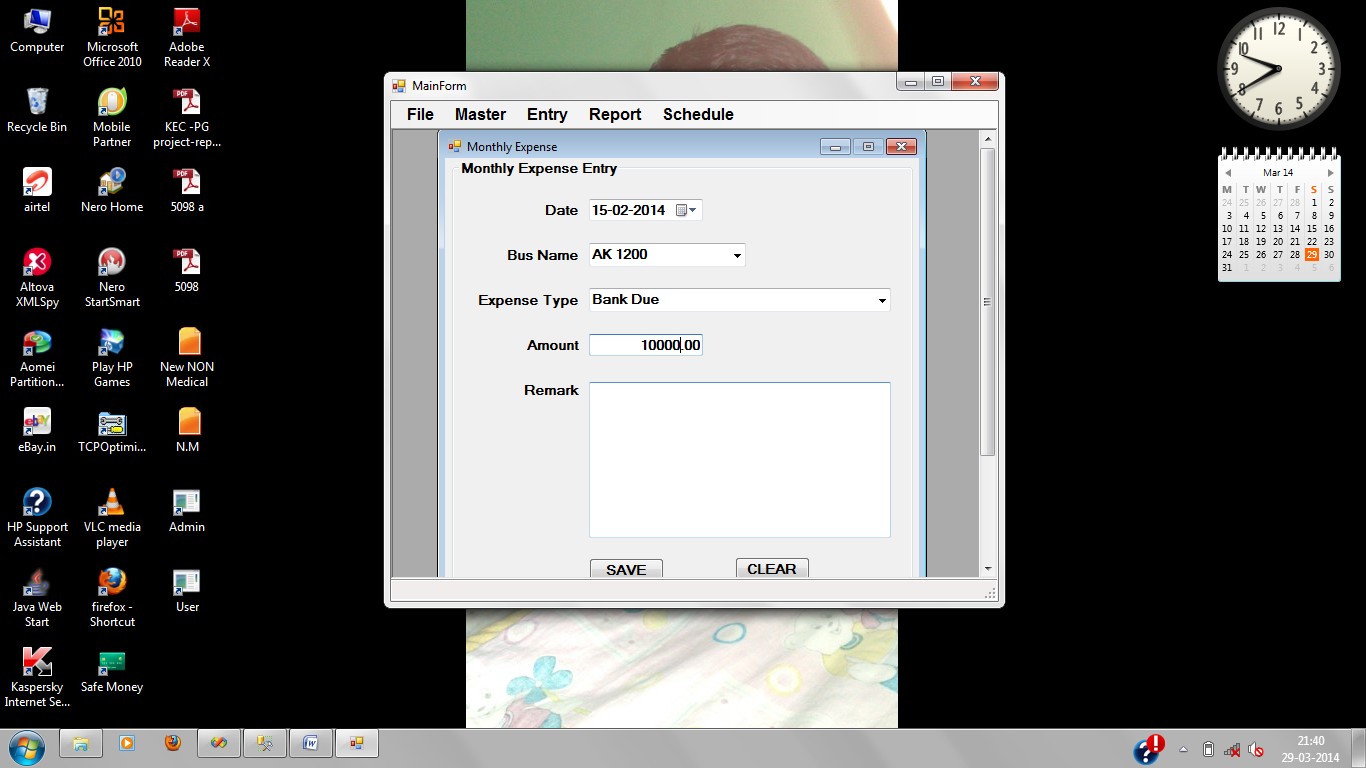
**Figure A2.1 Employee form**



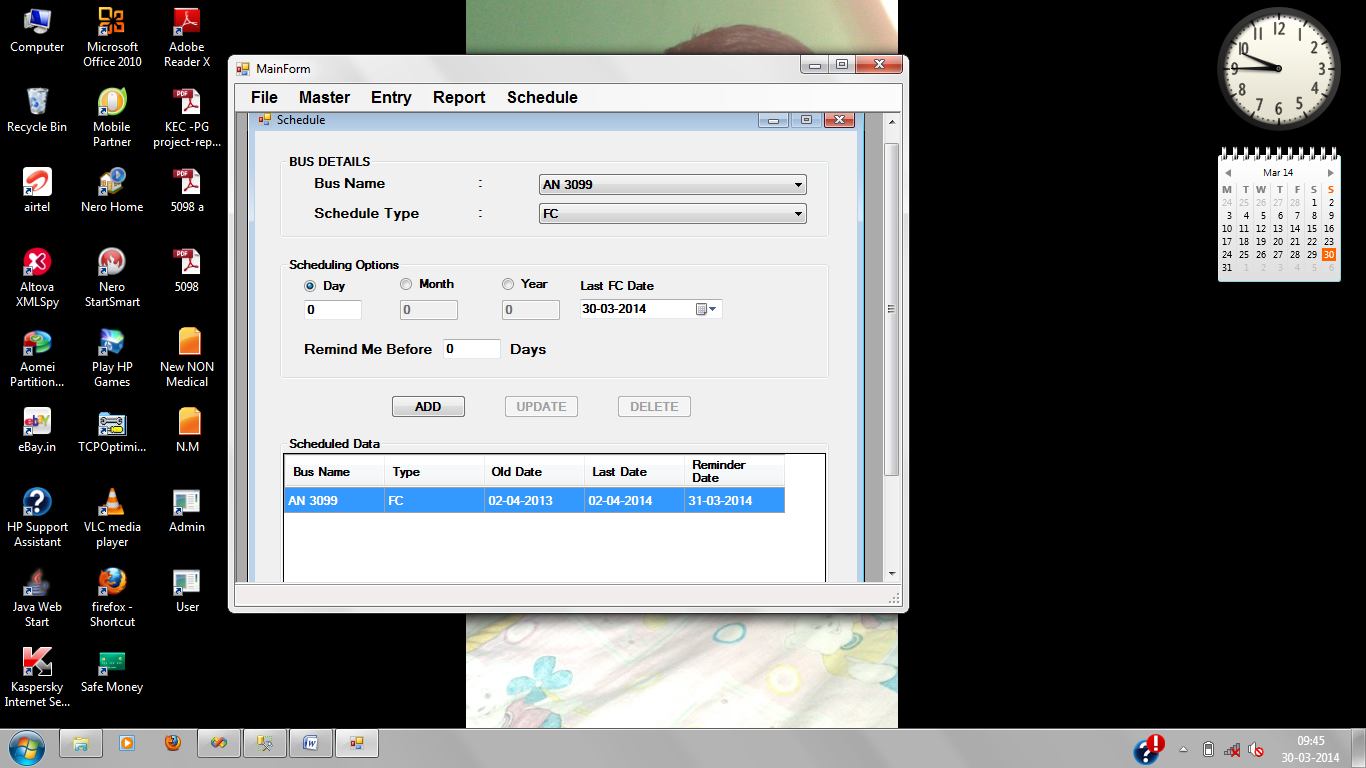
**Figure A2.2 Bus form**



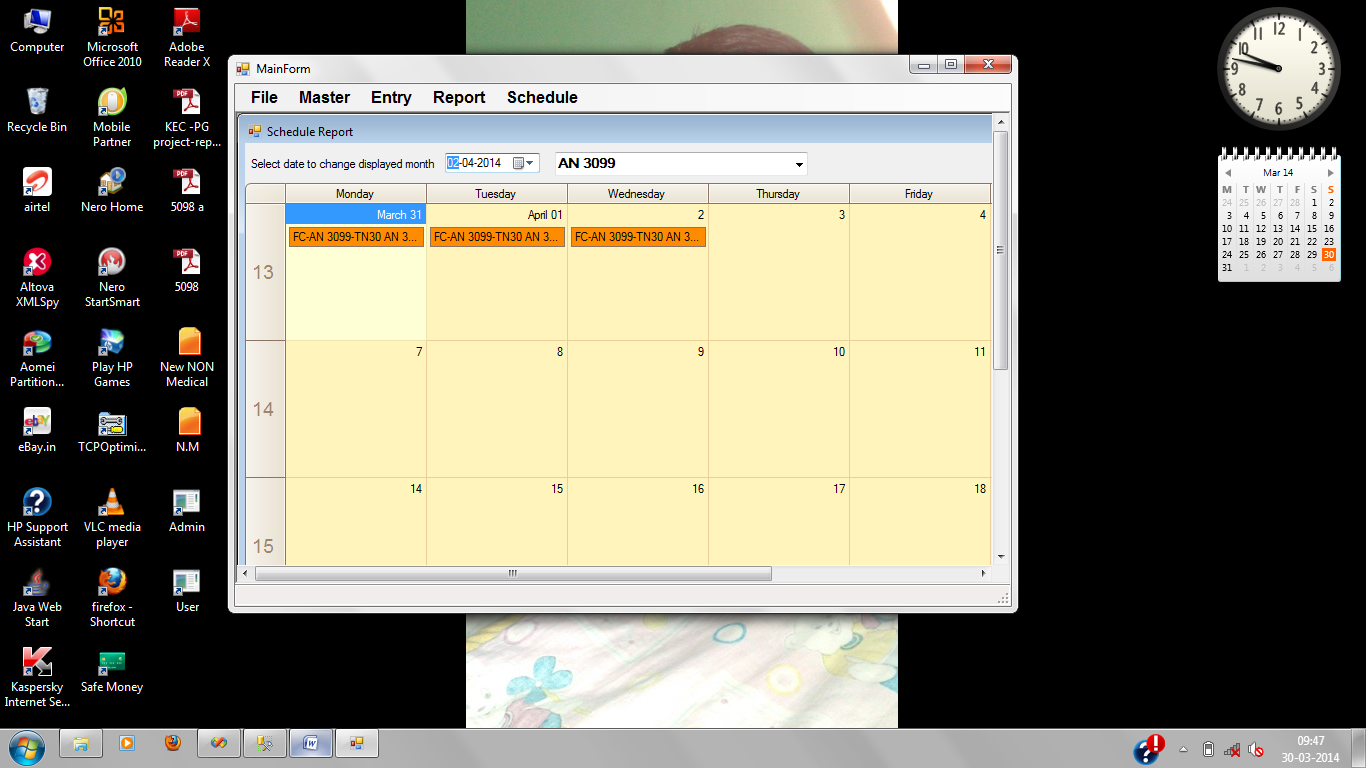
**Figure A2.3 Collection form**



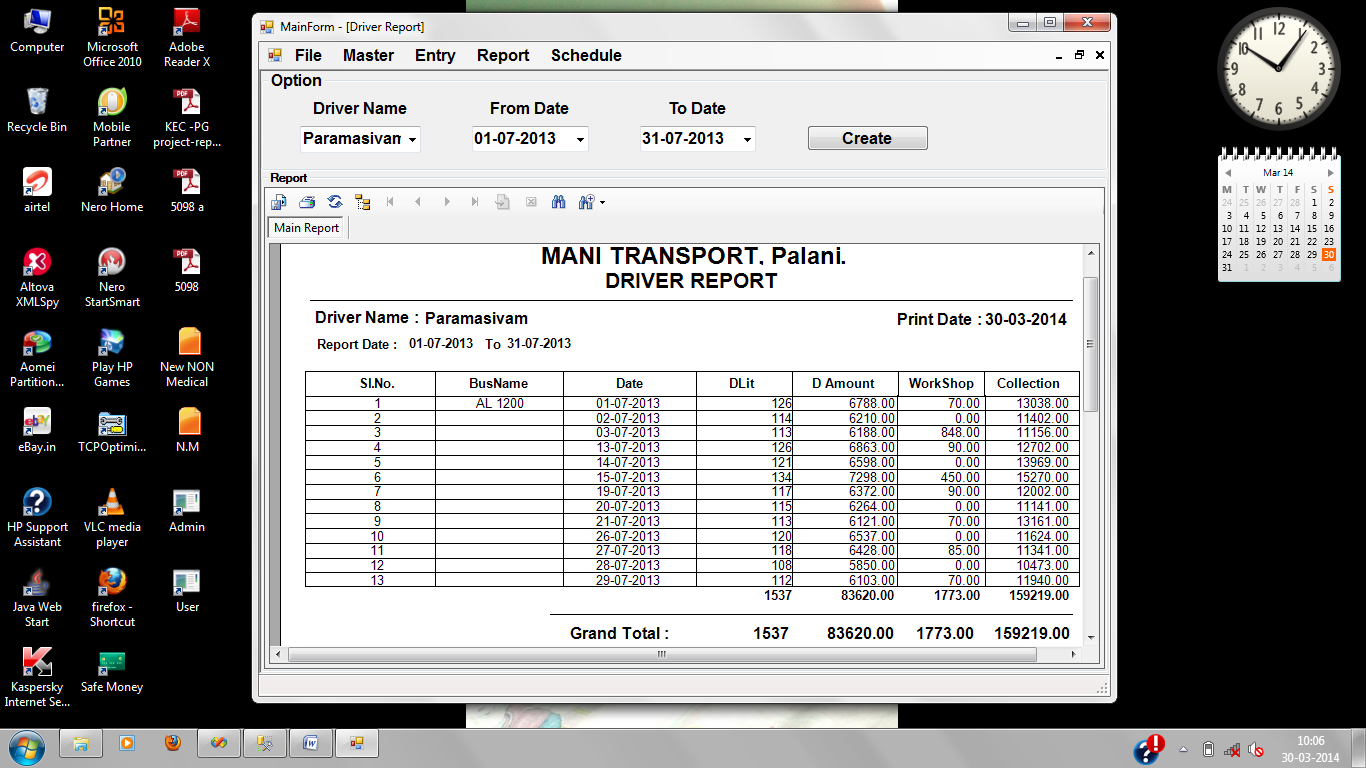
**Figure A2.4 Expense form**



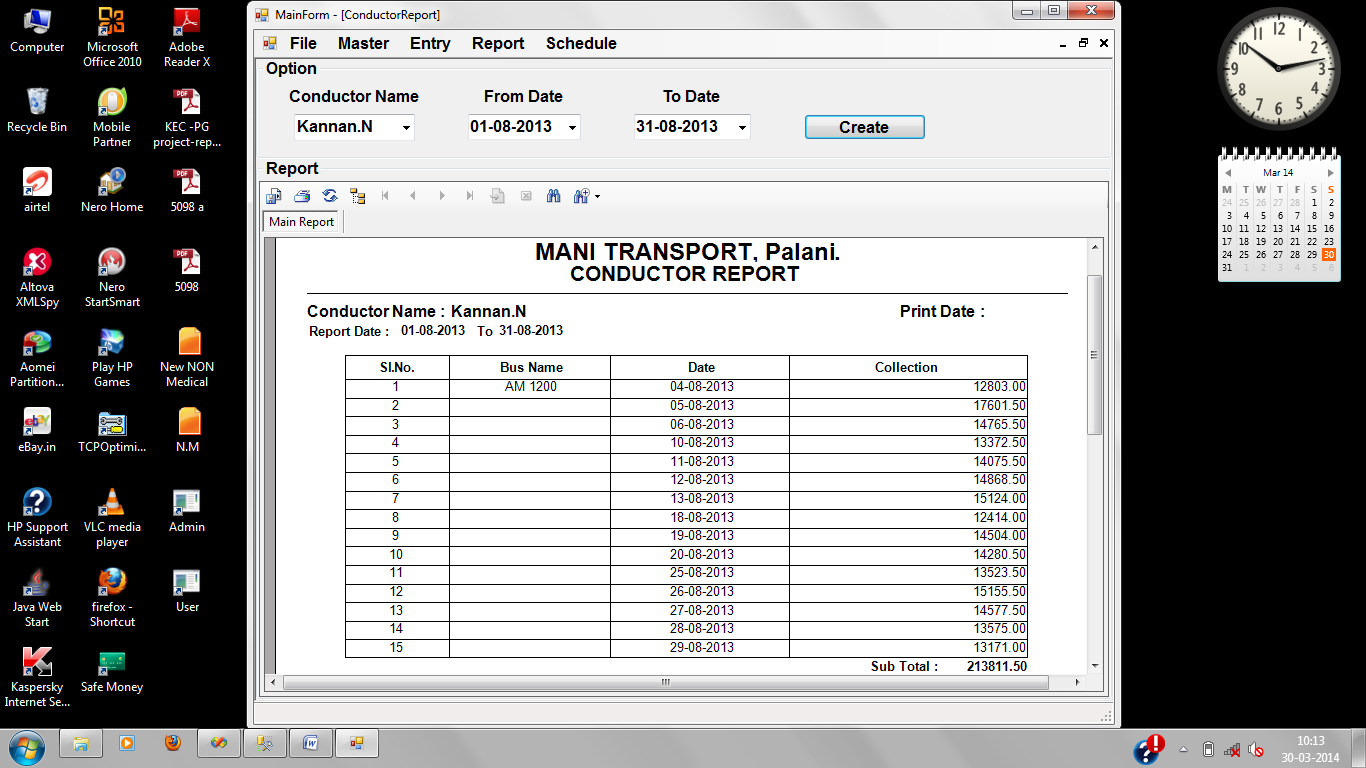
**Figure A2.5 Schedule form**



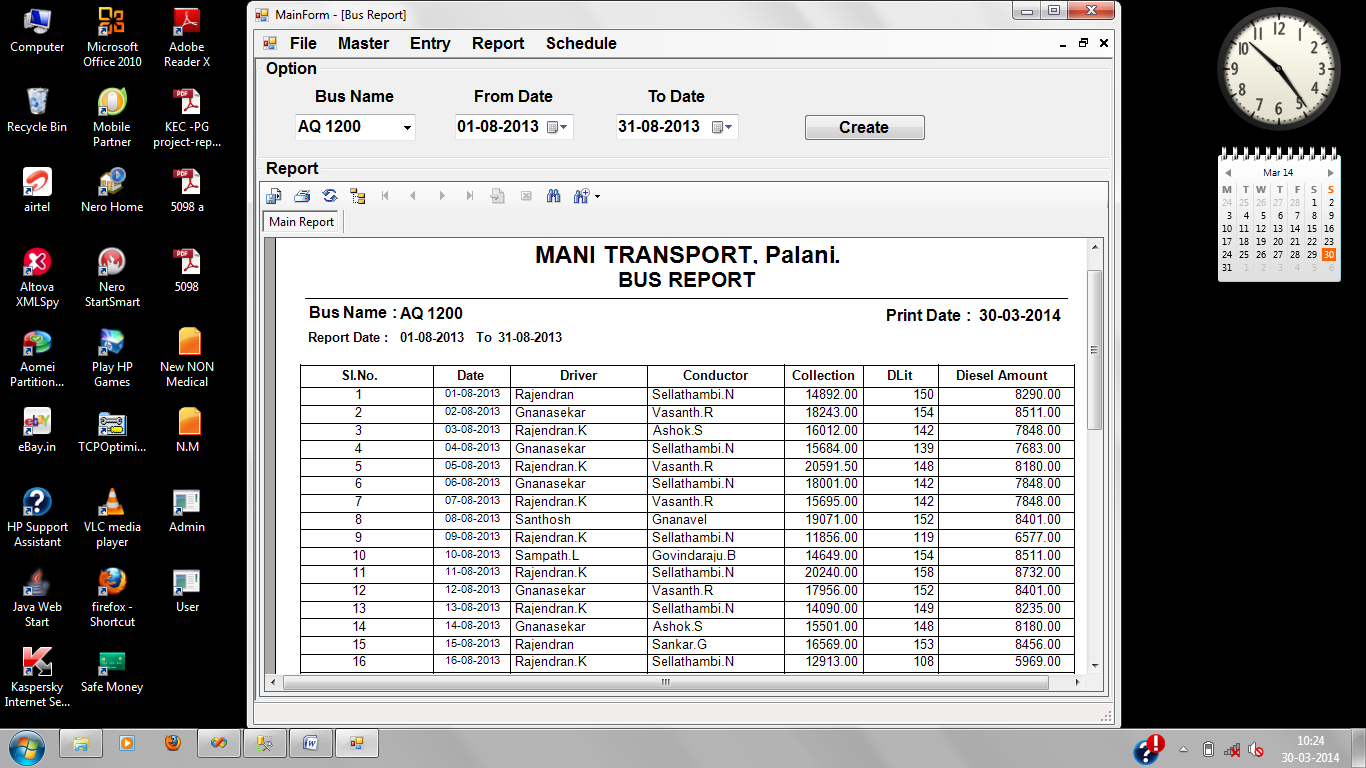
**Figure A2.6 Schedule alert messages**



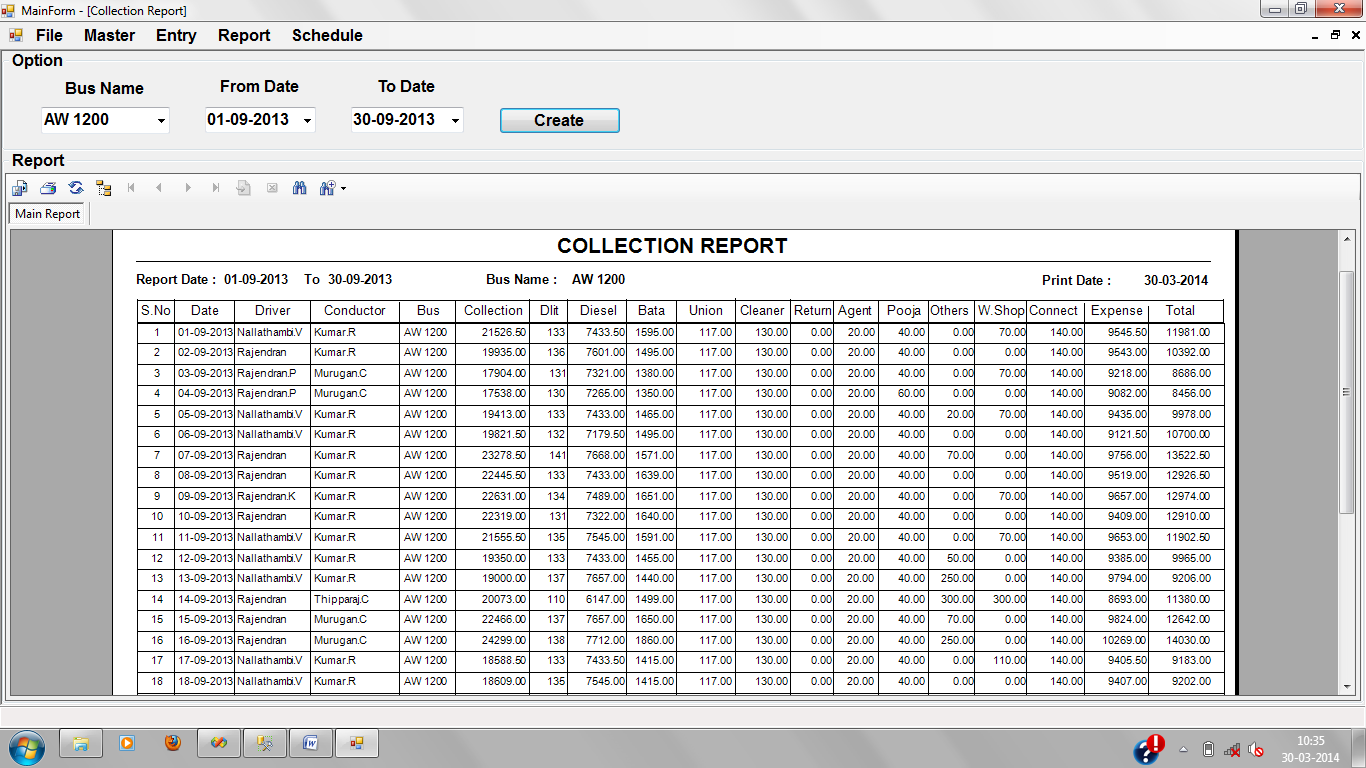
**Figure A2.7 Driver report**



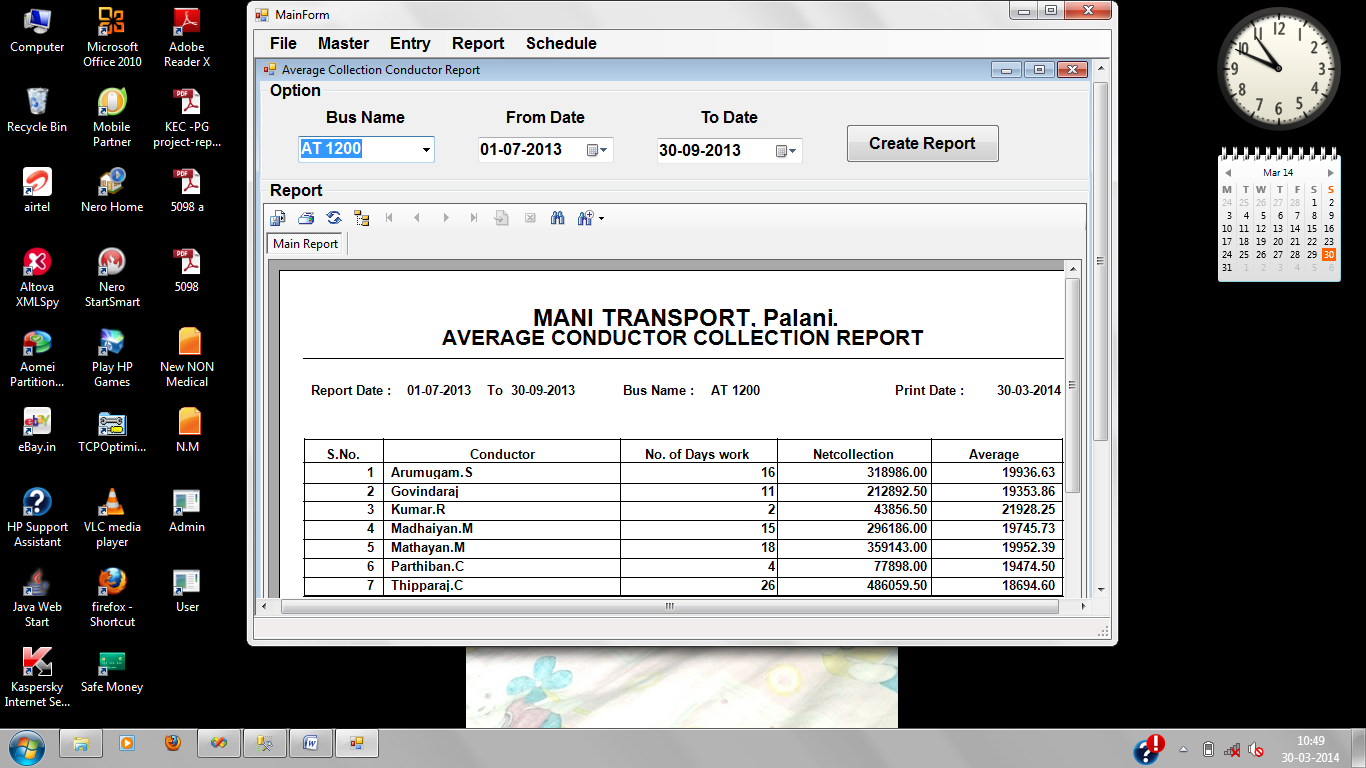
**Figure A2.8 Conductor report**



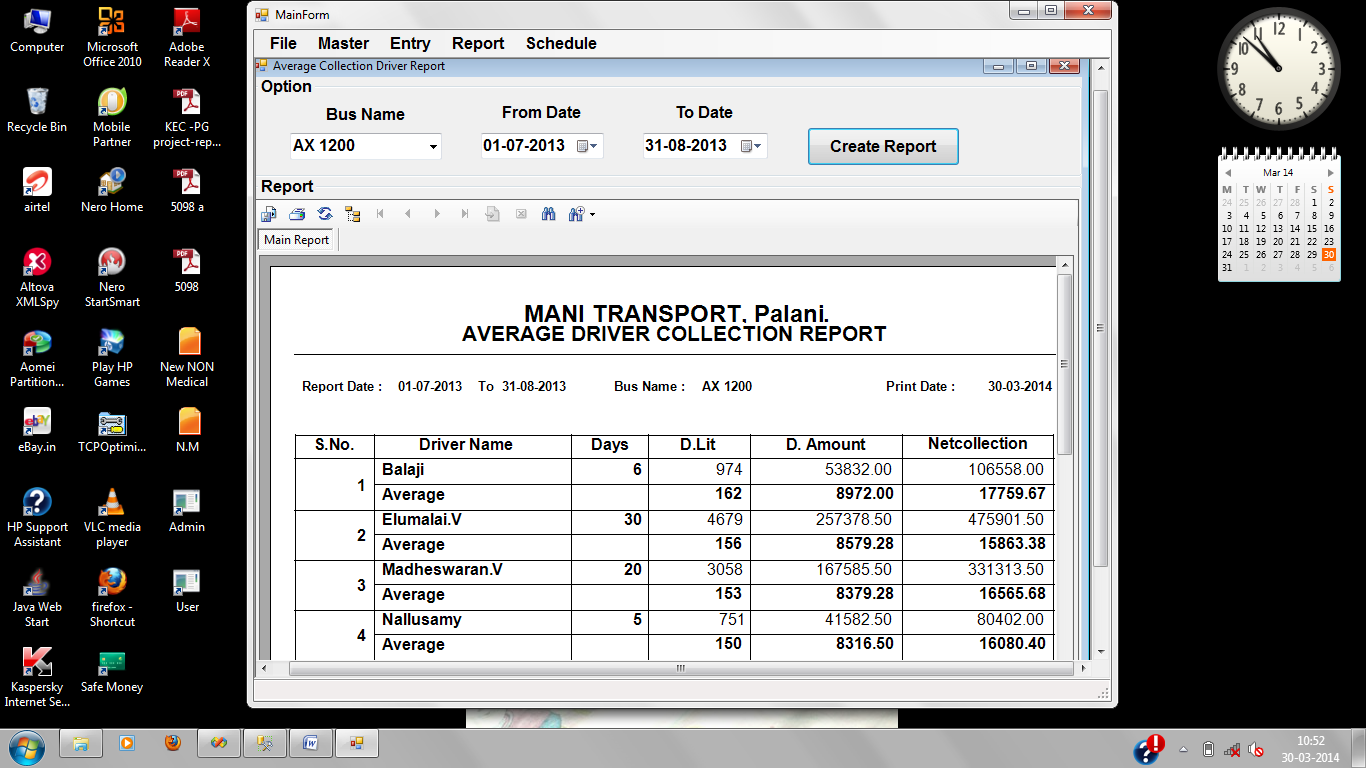
**Figure A2.9 Bus report**



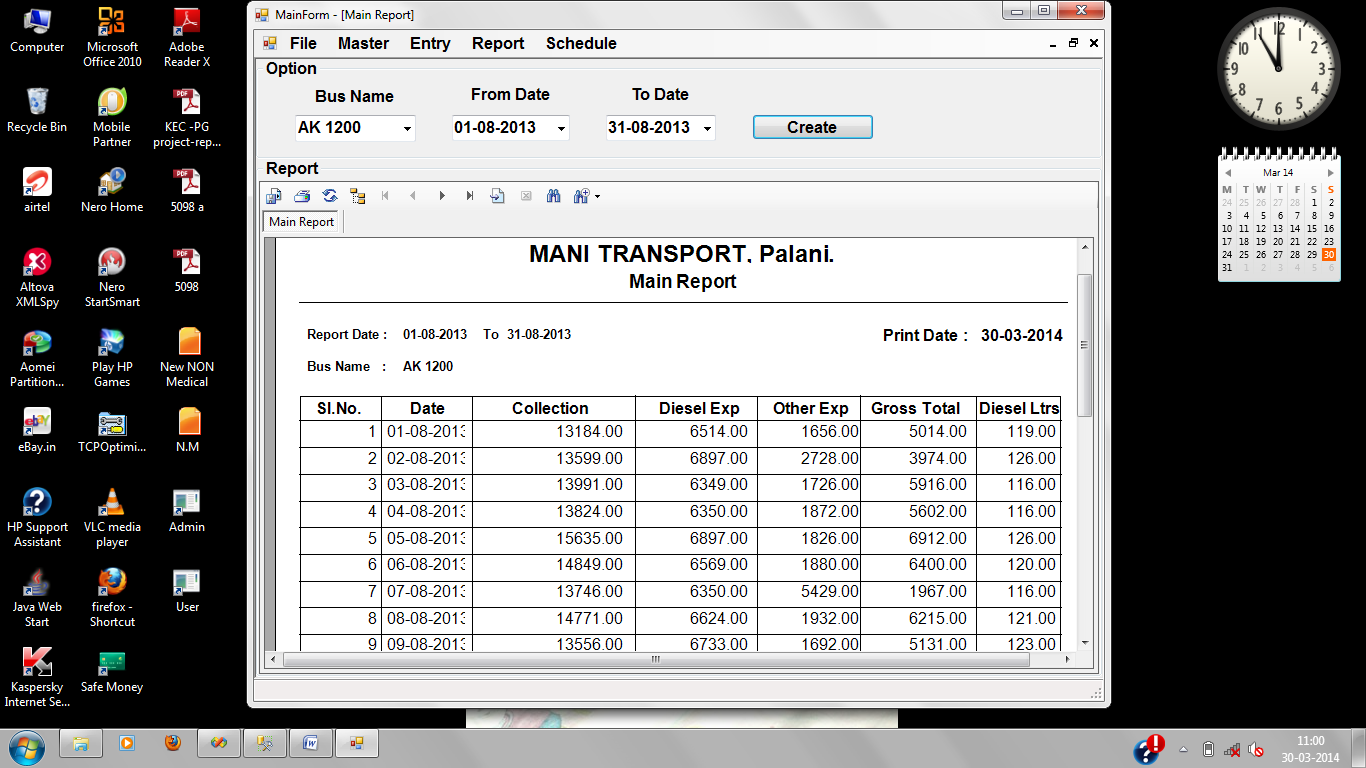
**Figure A2.10 Collection report**



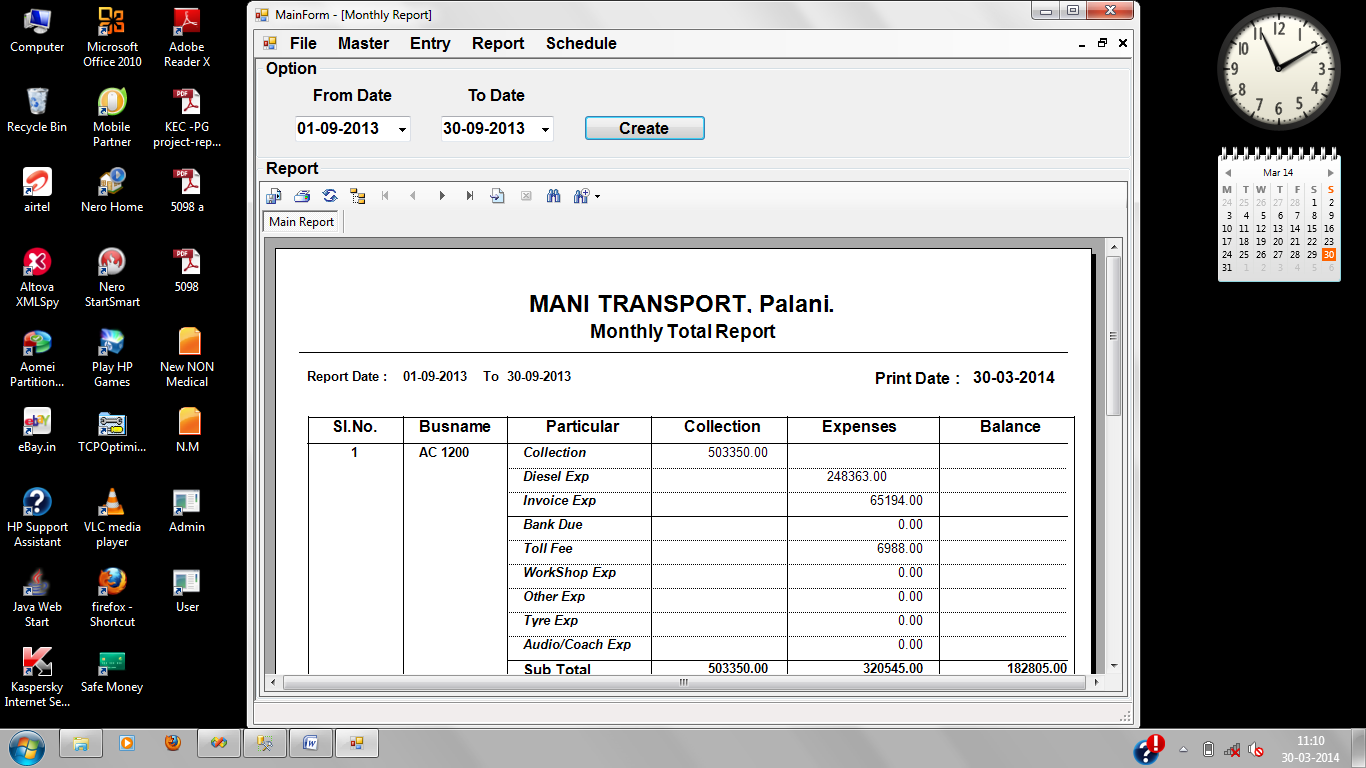
**Figure A2.11 Conductors average report**



**Figure A2.12 Drivers average report**



**Figure A2.13 Main report**



**Figure A2.14 Monthly report**

**REFERENCES**

* 1. Date C. J. Bnarosa (1990), “An Introduction to Database Systems”, 2nd Edition, Date C. J. Bnarosa Publishing House.
  2. Elias M. Awad (1998), “System Analysis and Design”, 2nd Edition, Techmedia Publication.
  3. Roger S. Pressman “Software Engineering” McGraw-Hill Publications.
  4. StevenHolzer, “The Visual basic .Net Programming”, DreamtechPress.
  5. [www.stackoverflow.com](http://www.stackoverflow.com).
  6. [www.sqlserver.com](http://www.sqlserver.com).
  7. [www.sqlserver.net/manual/en/sqlserver.](http://www.php.net/manual/en/sqlserver.)