REPORT

On

Vehicle Management System

Submitted by

Ajay Sunil (11606333)

Aravind Nallajerla (11606361)

Vishwakarma Krishna (11718906)

Section No: KE027

Programme Name: B. Tech Computer Science Engineering

School of Computer Science & Engineering

Lovely Professional University, Phagwara

1.) INTRODUCTION

Pack your bags, vehicle management system is a web based project and it has been developed in ASP, C#, MySQL and we can manage booking, vehicle, passenger, route, payment etc. from this project. The main objective to develop the vehicle management system is to overcome the manual errors and make a computerized system.

The 'Pack Your Bags' keeps the information about the Bus, Route, Employees and Passengers. It also keeps track the maintenance performed for different vehicles which are used for transportation.

The admin is the one who can add, edit or delete the details of Bus, Employee and Passenger. The admin may be the owner of the transportation organization or the manager of transportation department of a particular manufacturing company.

If any other vehicle is added to the count which already exists for the organization/department the details of the vehicle is added. The details include whether it is a new one or on service. Any employee is newly appointed or the existing employee is taken off both the details are maintained including their personal details and profession details.

The Passenger details too added to the system for the clarification. Also system allot the seat and the bus including the route to the database attached with the passenger.

1.2 PURPOSE OF THE PROJECT

Purpose of developing this application is to streamline the vehicle management process in the company and avoid any human errors. Tracking of vehicle information manually, especially when hundreds vehicles are there for the organization, is a tedious process. One of the major tasks is sending the vehicles for service. In manual process there won't be any provision to know that the vehicle should be sent for servicing, unless the vehicle record is verified manually.

1.3 PROBLEM IN EXISTING SYSTEM

- Managing huge Fleet information manually is a tedious and error prone task.
- In order to schedule vehicles as well as staff, we the scheduler should not how many vehicles are there on board and available for allocation.
- Keeping track of repair information is a must as some times vehicles might be referred for insurance.

  These entire things cannot be achieved in existing system.

1.4. SOLUTION OF THESE PROBLEMS

- In order to avoid the limitations in the existing system, the current system is being developed.
- All vehicle details will be automated along with the staff information.
- Scheduling of trips and repair information is being fully automated to overcome chaos in the system.

2.)FEASIBILITY REPORT

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

- Technical Feasibility
- Operation Feasibility
- Economic Feasibility

2.1 TECHINICAL FEASIBILITY:

Evaluating the technical feasibility is the trickiest part of a feasibility study. This is because, at this point in time, not too many detailed design of the system, making it difficult to access issues like performance, costs on (on account of the kind of technology to be deployed) etc.

A number of issues have to be considered while doing a technical analysis.

Understand the different technologies involved in the proposed system:

Before commencing the project, we have to be very clear about what are the technologies that are to be required for the development of the new system.

Find out whether the organization currently possesses the required technologies:

Is the required technology available with the organization?

If so is the capacity sufficient?

For instance –

"Will the current printer be able to handle the new reports and forms required for the new system?"

2.2 OPERATIONAL FEASIBILITY:

Proposed projects are beneficial only if they can be turned into information systems that will meet the organizations operating requirements. Simply stated, this test of feasibility asks if the system will work when it is developed and installed. Are there major barriers to Implementation? Here are questions that will help test the operational feasibility of a project:

Is there sufficient support for the project from management from users? If the current system is well liked and used to the extent that persons will not be able to see reasons for change, there may be resistance.

Are the current business methods acceptable to the user? If they are not, Users may welcome a change that will bring about a more operational and useful systems.

Has the user been involved in the planning and development of the project? Early involvement reduces the chances of resistance to the system and in General and increases the likelihood of successful project.

Since the proposed system was to help reduce the hardships encountered in the existing manual system, the new system was considered to be operational feasible.

2.3 ECONOMIC FEASIBILITY:

It refers to the benefits or Outcomes we are deriving from the product as compared to the total cost we are spending for developing the product. If the benefits are more or less the same as the older system, then it is not feasible to develop the product.

In the present system, the development of new product greatly enhances the accuracy of the system and cuts short the delay in the processing of Birth and Death application. The errors can be greatly reduced and at the same time providing a great level of security. Here we don't need any additional equipment except memory of required capacity.

No need for spending money on client for maintenance because the database used is web enabled database.

3.) SYSTEM ANALYSIS

3.1 MODULES

- Bus Details.
- Employee Details.
- Passenger Details.
- Bus Route.
- Payment.
- Booking Details.

Bus details

- This includes the company name of the bus, Type of bus, engine no, Route details.
- We can add or edit the details of the bus.

Employee Details

- The application will keep track of many details include Employee number, Name, personal information, License Information.
- It also allows you to edit your Employee Information. Under this module there will be facility to add, delete, Modify the information regarding Employee.

Passenger Details

- Passengers details are stored wisely to find the needed route for the passenger, and allot the seat in the respective bus.
- Passenger details can be edited and deleted.

4.) PROJECT LIFECYCLE

4.1 DESCRIPTION

The waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards (like a waterfall) through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach does not define the process to go back to the previous phase to handle changes in requirement. The waterfall approach is the earliest approach that was used for software development.

❖ HARDWARE REQUIREMENT:

- i3 Processor Based Computer or higher
- Memory: 2 GB
- Hard Drive: 50 GB
- Internet Connection

❖ SOFTWARE REQUIREMENT:

- Windows 7 or higher
- Visual Studio
- SQL Server
- Google Chrome Browser

❖ ADVANTAGES

- Time consuming.
- Cost efficiency.
- Easy to find applicants.
- Easy to find job.

❖ LIMITATION

- Increase in competitions.
- Needs active internet connection.

❖ APPLICATION

- Search module has been implemented to search vehicle, customer, booking.
- Vehicle Management System is an online based web application, from which users can easily manage booking, bus route details from the browser.
- Admin user will be able to track all the information of booking, vehicle, user etc.
- Admin user has the rights to edit, add, delete and update all the records of Drivers, Customer, vehicle, Type.

5.) INTRODUCTION TO .NET FRAMEWORK

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

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

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

applications to applications based on the latest innovations provided by ASP.NET, such as Web Forms and XML Web services.

The .NET Framework can be hosted by unmanaged components that load the common language runtime into their processes and initiate the execution of managed code, thereby creating a software environment that can exploit both managed and unmanaged features. The .NET Framework not only provides several runtime hosts, but also supports the development of third-party runtime hosts.

For example, ASP.NET hosts the runtime to provide a scalable, server-side environment for managed code. ASP.NET works directly with the runtime to enable Web Forms applications and XML Web services, both of which are discussed later in this topic.

Internet Explorer is an example of an unmanaged application that hosts the runtime (in the form of a MIME type extension). Using Internet Explorer to host the runtime enables you to embed managed components or Windows Forms controls in HTML documents. Hosting the runtime in this way makes managed mobile code (similar to Microsoft® ActiveX® controls) possible, but with significant improvements that only managed code can offer, such as semi-trusted execution and secure isolated file storage.

The following illustration shows the relationship of the common language runtime and the class library to your applications and to the overall system. The illustration also shows how managed code operates within a larger architecture.

FEATURES OF THE COMMON LANGUAGE RUNTIME

The common language runtime manages memory, thread execution, code execution, code safety verification, compilation, and other system services. These features are intrinsic to the managed code that runs on the common language runtime.

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform

file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

The runtime enforces code access security. For example, users can trust that an executable embedded in a Web page can play an animation on screen or sing a song, but cannot access their personal data, file system, or network. The security features of the runtime thus enable legitimate Internet-deployed software to be exceptionally featuring rich.

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers

Generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

In addition, the managed environment of the runtime eliminates many common software issues. For example, the runtime automatically handles object layout and manages references to objects, releasing them when they are no longer being used. This automatic memory management resolves the two most common application errors, memory leaks and invalid memory references.

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

While the runtime is designed for the software of the future, it also supports software of today and yesterday. Interoperability between managed and unmanaged code enables developers to continue to use necessary COM components and DLLs.

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the

system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

Finally, the runtime can be hosted by high-performance, server-side applications, such as Microsoft® SQL Server™ and Internet Information Services (IIS). This infrastructure enables you to use managed code to write your business logic, while still enjoying the superior performance of the industry's best enterprise servers that support runtime hosting.

.NET FRAMEWORK CLASS LIBRARY

The .NET Framework class library is a collection of reusable types that tightly integrate with the common language runtime. The class library is object oriented, providing types from which your own managed code can derive functionality. This not only makes the .NET Framework types easy to use, but also reduces the time associated with learning new features of the .NET Framework. In addition, third-party components can integrate seamlessly with classes in the .NET Framework.

For example, the .NET Framework collection classes implement a set of interfaces that you can use to develop your own collection classes. Your collection classes will blend seamlessly with the classes in the .NET Framework.

As you would expect from an object-oriented class library, the .NET Framework types enable you to accomplish a range of common programming tasks, including tasks such as string management, data collection, database connectivity, and file access. In addition to these common tasks, the class library includes types that support a variety of specialized development scenarios. For example, you can use the .NET Framework to develop the following types of applications and services:

- Console applications.
- Scripted or hosted applications.
- Windows GUI applications (Windows Forms).
- ASP.NET applications.

- XML Web services.
- Windows services.

For example, the Windows Forms classes are a comprehensive set of reusable types that vastly simplify Windows GUI development. If you write an ASP.NET Web Form application, you can use the Web Forms classes.

CLIENT APPLICATION DEVELOPMENT

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus, buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

Another kind of client application is the traditional ActiveX control (now replaced by the managed Windows Forms control) deployed over the Internet as a Web page. This application is much like other client applications: it is executed natively, has access to local resources, and includes graphical elements.

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications.

The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases, the underlying operating system does not support changing

these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

Unlike ActiveX controls, Windows Forms controls have semi-trusted access to a user's computer. This means that binary or natively executing code can access some of the resources on the user's system (such as GUI elements and limited file access) without being able to access or compromise other resources. Because of code access security, many applications that once needed to be installed on a user's system can now be safely deployed through the Web. Your applications can implement the features of a local application while being deployed like a Web page.

6.) DATA FLOW DIAGRAMS

A data flow diagram is graphical tool used to describe and analyze movement of data through a system. These are the central tool and the basis from which the other components are developed. The transformation of data from input to output, through processed, may be described logically and independently of physical components associated with the system. These are known as the logical data flow diagrams. The physical data flow diagrams show the actual implements and movement of data between people, departments and workstations. A full description of a system actually consists of a set of data flow diagrams. Using two familiar notations Yourdon, Gane and Sarson notation develops the data flow diagrams. Each component in a DFD is labeled with a descriptive name. Process is further identified with a number that will be used for identification purpose. The development of DFD'S is done in several levels. Each process in lower level diagrams can be broken down into a more detailed DFD in the next level. The lop-level diagram is often called context diagram. It consists a single process bit, which plays vital role in studying the current system. The process in the context level diagram is exploded into other process at the first level DFD.

The idea behind the explosion of a process into more process is that understanding at one level of detail is exploded into greater detail at the next level. This is done until further explosion is necessary and an adequate amount of detail is described for analyst to understand the process.

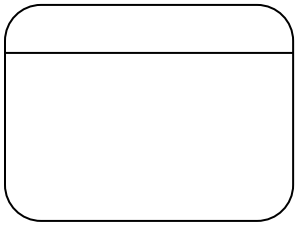
Larry Constantine first developed the DFD as a way of expressing system requirements in a graphical from, this lead to the modular design.

A DFD is also known as a "bubble Chart" has the purpose of clarifying system requirements and identifying major transformations that will become programs in system design. So it is the starting point of the design to the lowest level of detail. A DFD consists of a series of bubbles joined by data flows in the system.

DFD SYMBOLS:

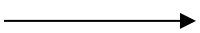
In the DFD, there are four symbols

1. A square defines a source(originator) or destination of system data
2. An arrow identifies data flow. It is the pipeline through which the information flows
3. A circle or a bubble represents a process that transforms incoming data flow into outgoing data flows.
4. An open rectangle is a data store, data at rest or a temporary repository of data



Process that transforms data flow.

Source or Destination of data



Data flow

Data Store

CONSTRUCTING A DFD:

Several rules of thumb are used in drawing DFD'S:

1. Process should be named and numbered for an easy reference. Each name should be representative of the process.
2. The direction of flow is from top to bottom and from left to right. Data traditionally flow from source to the destination although they may flow back to the source. One way to indicate this is to draw long flow line back to a source. An alternative way is to repeat the source symbol as a destination. Since it is used more than once in the DFD it is marked with a short diagonal.
3. When a process is exploded into lower level details, they are numbered.
4. The names of data stores and destinations are written in capital letters. Process and dataflow names have the first letter of each work capitalized

A DFD typically shows the minimum contents of data store. Each data store should contain all the data elements that flow in and out.

Questionnaires should contain all the data elements that flow in and out. Missing interfaces redundancies and like is then accounted for often through interviews.

SAILENT FEATURES OF DFD'S

1. The DFD shows flow of data, not of control loops and decision are controlled considerations do not appear on a DFD.
2. The DFD does not indicate the time factor involved in any process whether the dataflow take place daily, weekly, monthly or yearly.
3. The sequence of events is not brought out on the DFD.

TYPES OF DATA FLOW DIAGRAMS

1. Current Physical
2. Current Logical

3. New Logical
4. New Physical

CURRENT PHYSICAL:

In Current Physical DFD process label include the name of people or their positions or the names of computer systems that might provide some of the overall system-processing label includes an identification of the technology used to process the data. Similarly, data flows and data stores are often labels with the names of the actual physical media on which data are stored such as file folders, computer files, business forms or computer tapes.

CURRENT LOGICAL:

The physical aspects at the system are removed as much as possible so that the current system is reduced to its essence to the data and the processors that transforms them regardless of actual physical form.

NEW LOGICAL:

This is exactly like a current logical model if the user were completely happy with the user were completely happy with the functionality of the current system but had problems with how it was implemented typically through the new logical model will differ from current logical model while having additional functions, absolute function removal and inefficient flows recognized.

NEW PHYSICAL:

The new physical represents only the physical implementation of the new system.

RULES GOVERNING THE DFD'S

PROCESS

1) No process can have only outputs.
2) No process can have only inputs. If an object has only inputs than it must be a sink.

3) A process has a verb phrase label.

DATA STORE

1) Data cannot move directly from one data store to another data store, a process must move data.
2) Data cannot move directly from an outside source to a data store, a process, which receives, must move data from the source and place the data into data store
3) A data store has a noun phrase label.

SOURCE OR SINK

The origin and / or destination of data.

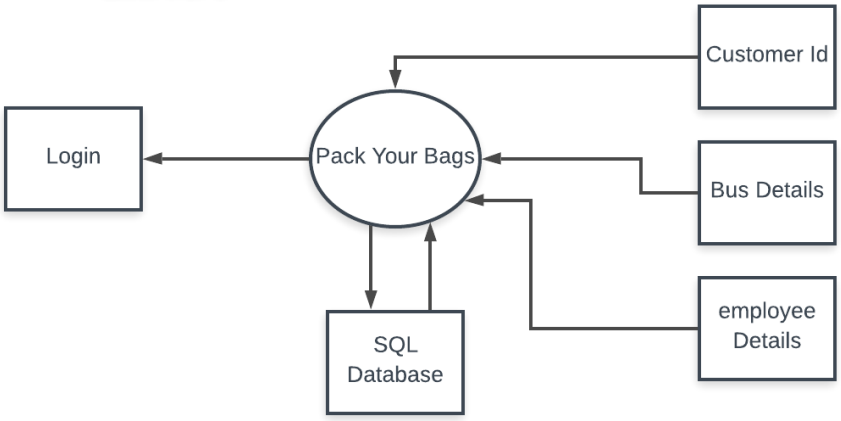
1) Data cannot move direly from a source to sink it must be moved by a process
2) A source and /or sink has a noun phrase land

DATA FLOW

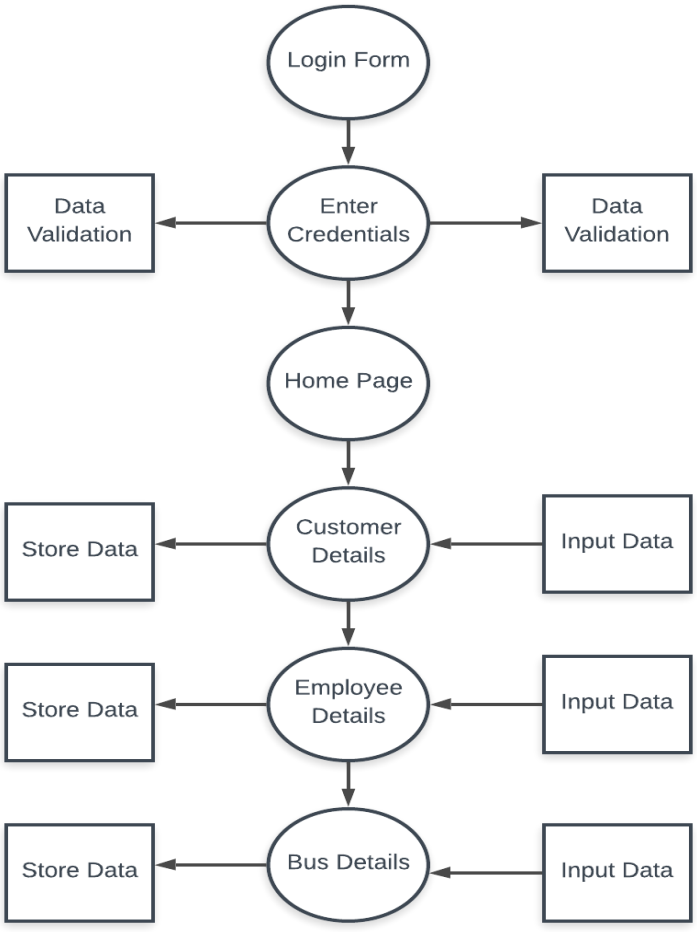
1) A Data Flow has only one direction of flow between symbols. It may flow in both directions between a process and a data store to show a read before an update. The latter is usually indicated however by two separate arrows since these happen at different type.
2) A join in DFD means that exactly the same data comes from any of two or more different processes data store or sink to a common location.
3) A data flow cannot go directly back to the same process it leads. There must be at least one other process that handles the data flow produce some other data flow returns the original data into the beginning process.
4) A Data flow to a data store means update (delete or change).
5) A data Flow from a data store means retrieve or use.

A data flow has a noun phrase label more than one data flow noun phrase can appear on a single arrow as long as all of the flows on the same arrow move together as one package.

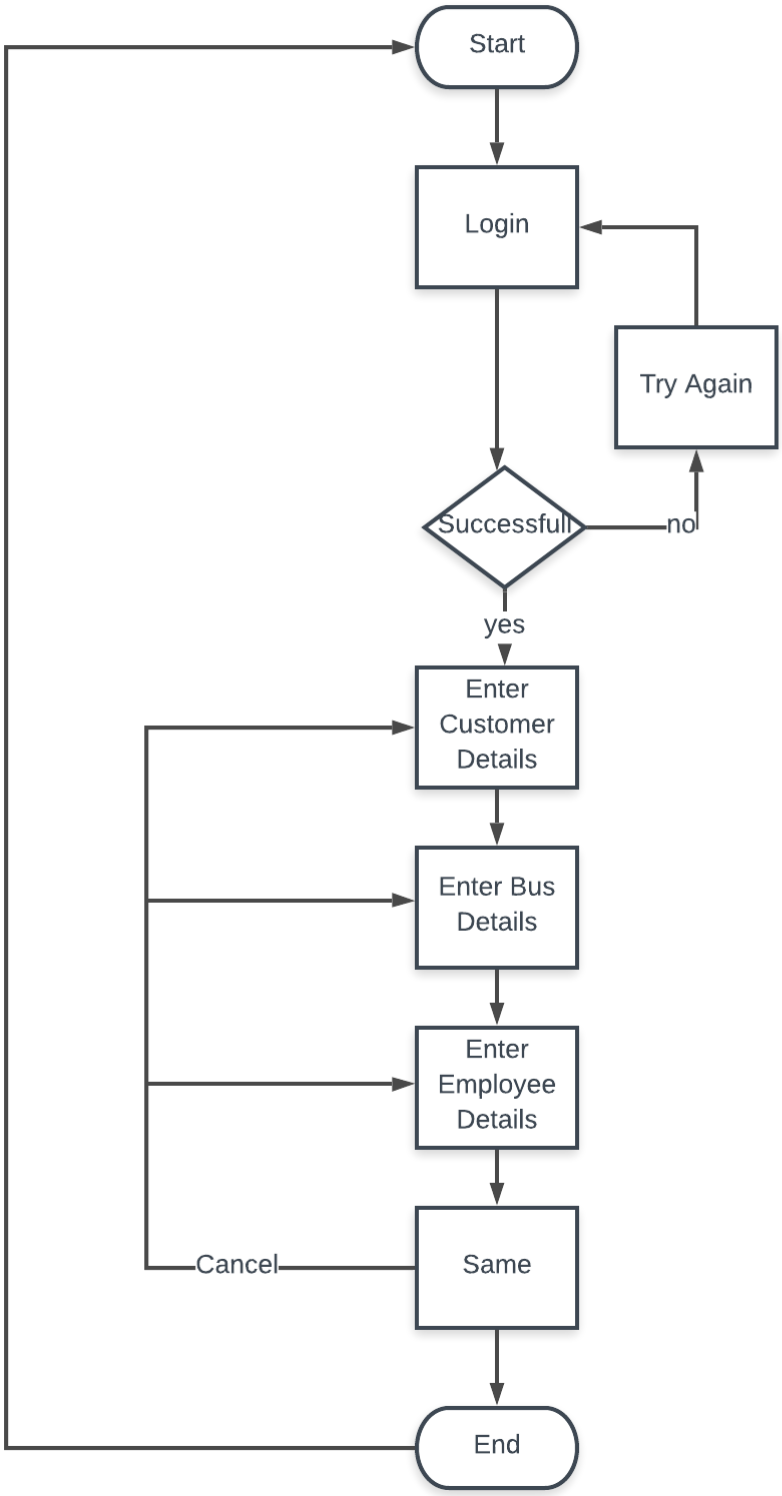
Level 0 DFD



Level 1 DFD

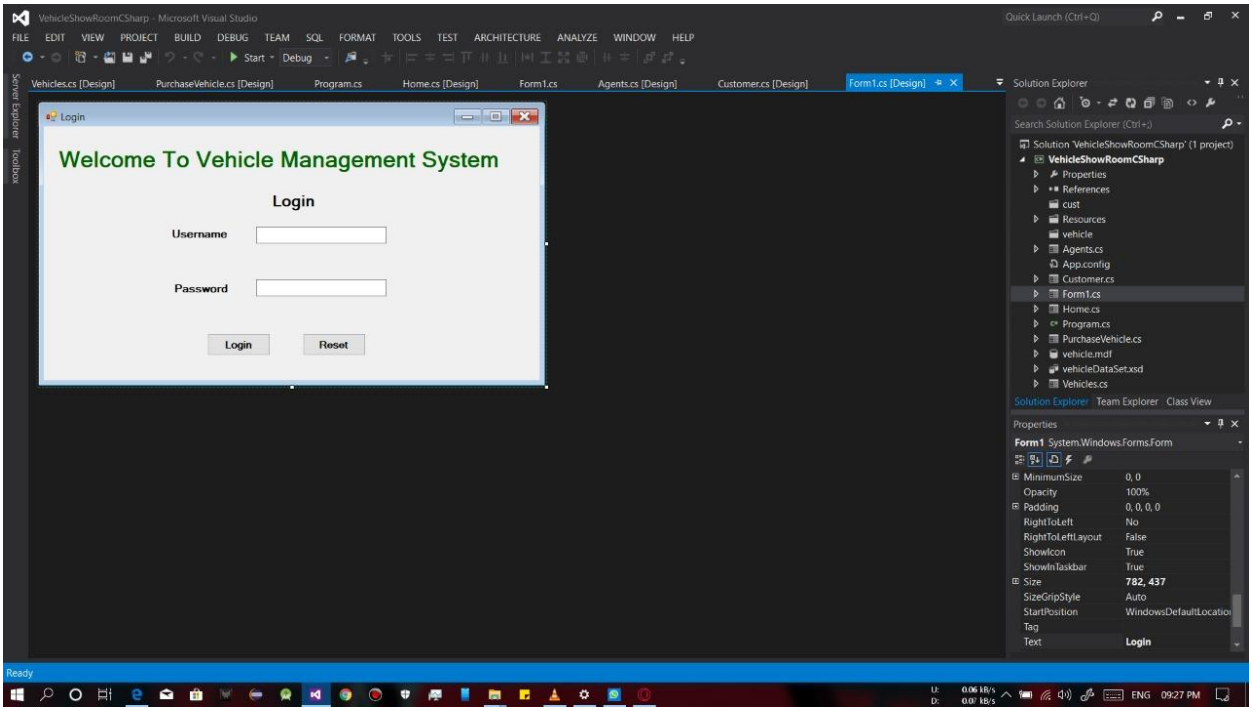


FLOW CHART FOR
PACK YOUR BAGS

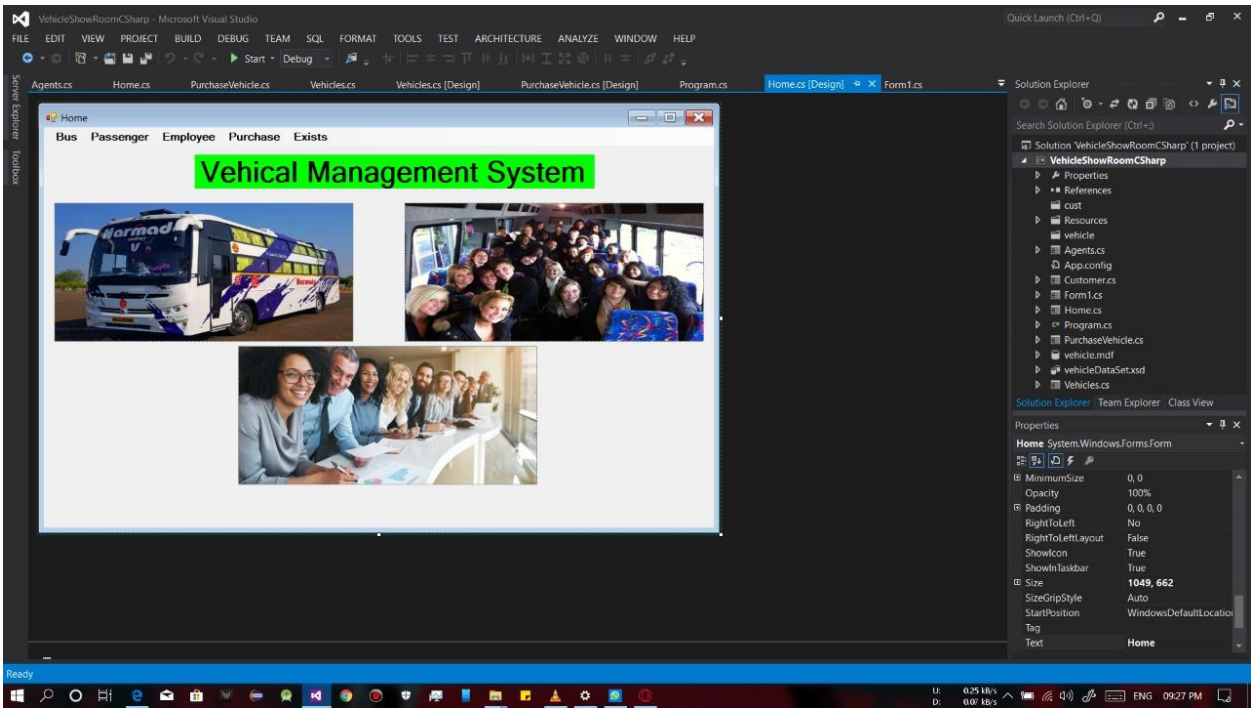


7.) SCREENSHOTS

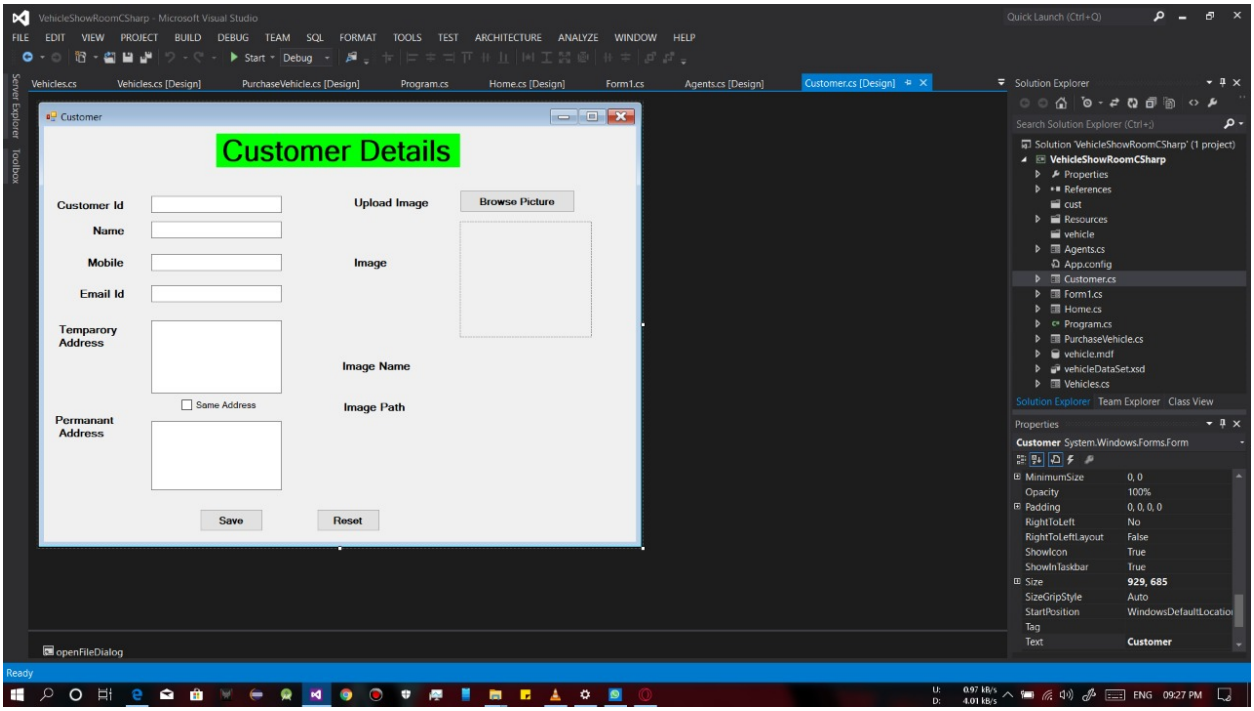
Login Page



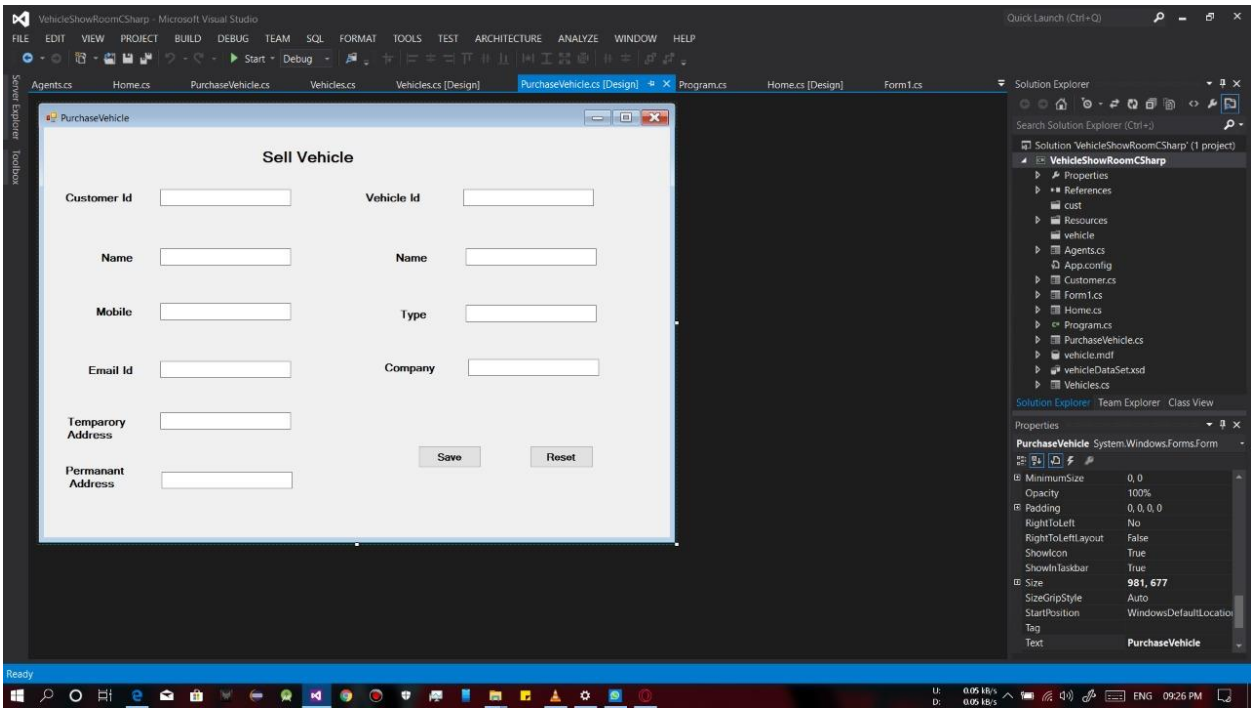
Home Page



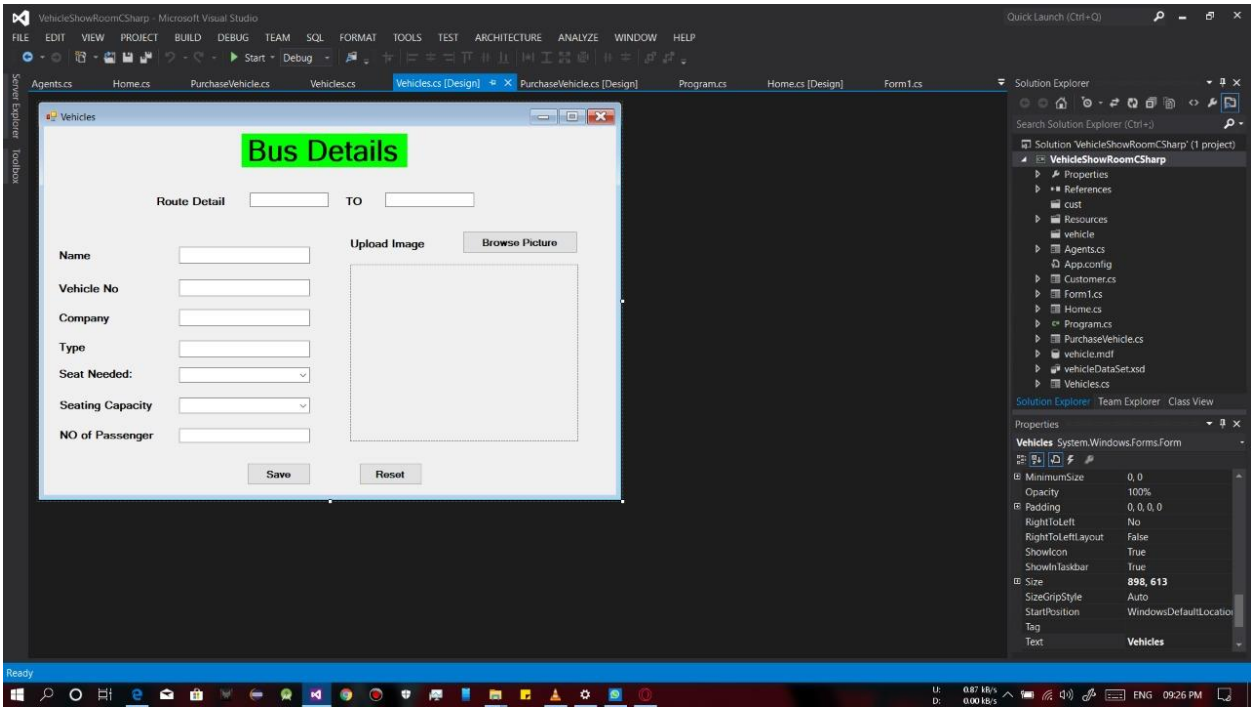
Customer Details



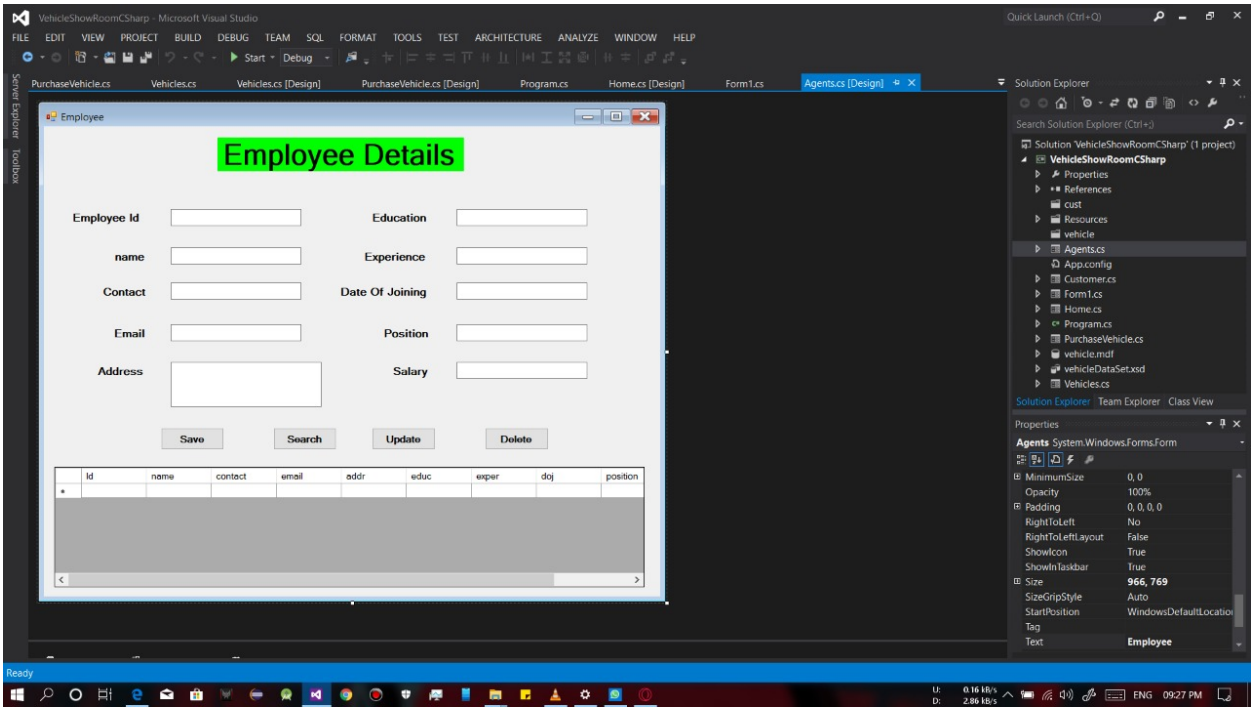
Sell Vehicle page



Bus Details



Employee Details



CONCLUSION

It has been a great pleasure for me to work on this exciting and challenging project. This project proved good for me as it provided practical knowledge of not only programming in ASP.NET windows based application, but also about all handling procedure related with **'Pack Your Bags'**. It also provides knowledge about the latest technology used in developing client server technology that will be great demand in future. This will provide better opportunities and guidance in future in developing projects independently.

This project offers user to enter the data through simple and interactive forms. This is very helpful for the client to enter the desired information through so much simplicity.

The user is mainly more concerned about the validity of the data, whatever he is entering. There are checks on every stages of any new creation, data entry or updating so that the user cannot enter the invalid data, which can create problems at later date.

Sometimes the user finds in the later stages of using project that he needs to update some of the information that he entered earlier. There are options for him by which he can update the records. Moreover, there is restriction for his that he cannot change the primary data field. This keeps the validity of the data to longer extent.

User is provided the option of monitoring the records he entered earlier. He can see the desired records with the variety of options provided by him.

BIBLIOGRAPHY

- **FOR .NET INSTALLATION**

  www.support.mircosoft.com

- **FOR DEPLOYMENT AND PACKING ON SERVER**

  www.developer.com

  www.15seconds.com

- **FOR SQL**

  www.msdn.microsoft.com

- **FOR ASP.NET**

  www.msdn.microsoft.com/net/quickstart/aspplus/default.com

  www.asp.net

  www.fmexpense.com/quickstart/aspplus/default.com

  www.asptoday.com

  www.aspfree.com

  www.4guysfromrolla.com/index.aspx

  https://shsu-ir.tdl.org/shsu-ir/bitstream/handle/20.500.11875/1164/0781.pdf?sequence=1

  https://ieeexplore.ieee.org/document/6208293/

  https://ieeexplore.ieee.org/document/4679917/