Abstract

Objective:-

            This system “Vehicle services” allows providing vehicle for service. If customer got any problem with his vehicle based on guarantee card they are providing services

            The real power of this project lies not in direct selling of products, but in the creation of tighter relationships with customers and delivering of a high level of service and support, which in turn improves organization sales and its goodwill. A service organization is a business entity that takes care of servicing a customer instrument in the after sales domain. As the number of customers and size of operations increases, the organization divides the geographical area into service areas and branch locations, to allow Engineers to be more responsive to the customer-needs.

 This project contains 3 modules namely :-

Ø  User Registration,

Ø  Service Charges,

Ø  Submission module,

User Creation:-

                   In this module  we are Registration  the  username ,password ,phno  and  vehicle no ,warranty and branch no’ are  user creations.

Service Charges:-

               In this module we are Registered the Customer Name, Vehicle no, Parts  and Servicing Charges as well as Paying the money.

Submission Module;-

                      In this module the dealer is handover to the Customer and before registering the  Customername,Vehicleno and phone.

Maintenance Order Registration and Warranties

INDEX

1. INTRODUCTION

2. ANALYSIS

2.1 SYSTEM ANALYSIS

2.2 SYSTEM SPECIFICATIONS

3. DESIGN APPROACH

3.1 INTRODUCTION TO DESIGN

3.2 UML DIAGRAMS

3.3 DATA FLOW DIAGRAMS

3.4 E-R DIAGRAMS

4. PROJECT MODULES

5. IMPLEMENTATION

4.1 CONCEPTS AND TECHNIQUES

4.2 TESTING

4.2.1 TEST CASES

6. OUTPUT SCREENS

7. CONCLUSION

8. FUTURE ENHANCEMENTS

9. BIBILIOGRAPHY

INTRODUCTION

More than two –third of  automotive customers  indicate that  Service

Convenience is a determining in selecting a brand  or purchasing from a Specific

dealership .Consequently, Customer Service  and service management is of vital relevance to ensure ongoing Customer loyalty and retention and ultimately, dealer Profitability.

Dealer Business Management  enables efficient Service order Processing and billing designed for any given number of orders per day .it includes Service  requests  and Scheduling ,Optimization of techicans,tools , and parts as well as their deployment and scheduling .User friendly interfaces provides ease of use and the integration technology ensure a seamless and smooth business process into OEM channel System, including ,job and Package Catalogs, Vehicle history files ,Warranty Systems, etc. Service monitoring and analysis increases the Visibility in Overall fixed Operations and helps increases service Capacity utilization ,efficiency and decreased operating cost.

Recreation Vehicle service technician inspect ,test, service, and replace every system installed in a recreation Vehicle with the exception  of the dry.

Both intervals are equally important for properly marinating your Vehicle  Remember all Toyota dealerships offer a broad range  of  Parts and Service.

The Driver and Vehicle Licensing Agency ,the Driving Standards  Agency and the Vehicle and Operator Service Agency Provide  Services for 42 million drivers.

The Vehicle  and Operator Services Agency(VOSA) Provides a range of licensing, testing and enforcement services with the aim of  improving the roadworthiness standards of Vehicles ensuring the Compliance of operators and drivers , and Supporting the independent Traffic  Commissioner.

SYSTEM ANALYSIS: - VEHICLE SERVICE MANAGEMENT SYSTEM PROJECT

1. Existing System

. Existing system is a manual one in which users are maintaining books to store the information like product details, Distributors details, purchases, sales details and accounts for every month. It is very difficult to maintain historical data.

z

The following are the disadvantages of the existing system

   It is difficult to maintain important information in books.

   More manual hours need to generate required reports.

   It is tedious to manage historical data which needs much space to keep all the previous years’ ledgers, books etc.

   Daily sales and purchases details must be entered into books are very difficult to maintain.

2. Proposed System

The DISTRIBUTORS MANAGEMENT TOOL is a software application which avoids more manual hours that need to spend in record keeping and generating reports. This application keeps the data in a centralized way which is available to all the users simultaneously. It is very easy to manage historical data in database. No specific training is required for the distributors to use this application. They can easily use the tool that decreases manual hours spending for normal things and hence increases the performance. It is very easy to record the information of online sales and purchases in the databases.

3. Objective of the System

            The objective of the Vehicle Services  is to provide better information for the users of this system for better results for their maintainence in the product details that is sales, purchases and stock.

                         System Specifications

Hardware Requirements:-

·         Pentium-IV(Processor).

·         256 MB Ram

·         512 KB Cache Memory

·         Hard disk 10 GB

·         Microsoft Compatible 101 or more Key Board

Software Requirements: -

Operating System :           Windows

Programming  language:   MS.NET

Web-Technology:               ASP.NET

Front-End:                          C#.NET

Back-End:                           SQLSERVER 2000

Web Server:                       IIS5.1

INTRODUCTION:

Design is the first step in the development phase for any techniques and principles for the purpose of defining a device, a process or system in sufficient detail to permit its physical realization.  
Once the software requirements have been analyzed and specified the software design involves three technical activities - design, coding, implementation and testing that are required to build and verify the software.  
The design activities are of main importance in this phase, because in this activity, decisions ultimately affecting the success of the software implementation and its ease of maintenance are made. These decisions have the final bearing upon reliability and maintainability of the system. Design is the only way to accurately translate the customer’s requirements into finished software or a system.  
Design is the place where quality is fostered in development. Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data.

UML Diagrams:

Actor:

            A coherent set of roles that users of use cases play when interacting with the use `cases.

|  |
| --- |
|  |
|  |  |

 Use case:

                              A description of sequence of actions, including variants, that a system performs that yields an observable result of value of an actor.

UML stands for Unified Modeling Language. UML is a language for specifying, visualizing and documenting the system. This is the step while developing any product after analysis. The goal from this is to produce a model of the entities involved in the project which later need to be built. The representation of the entities that are to be used in the product being developed need to be designed.

                        There are various kinds of methods in software design:

They are as follows:

Ø  Use case Diagram

Ø  Sequence Diagram

Ø  Collaboration Diagram

Ø  Activity Diagram

Ø  State chat Diagram

USE CASE DIAGRAMS:

Use case diagrams model behavior within a system and helps the developers understand of what the user require. The stick man represents what’s called an actor.

Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can’t do.  
  
Use case diagram consists of use cases and actors and shows the interaction between the use case and actors.

·         The purpose is to show the interactions between the use case and actor.

·         To represent the system requirements from user’s perspective.

·         An actor could be the end-user of the system or an external system.

USE CASE DIAGRAM:

A Use case is a description of set of sequence of actions.  Graphically it is rendered as an ellipse with solid line including only its name.  Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship.  It is an association between the use cases and actors.  An actor represents a real-world object.  Primary Actor – Sender, Secondary Actor Receiver.

SEQUENCE DIAGRAM:  
  
Sequence diagram and collaboration diagram are called INTERACTION DIAGRAMS. An interaction diagram shows an interaction, consisting of set of objects and their relationship including the messages that may be dispatched among them.

A sequence diagram is an introduction that empathizes the time ordering of messages. Graphically a sequence diagram is a table that shows objects arranged along the X-axis and messages ordered in increasing time along the Y-axis

COLLABORATION DIAGRAM:

A collaboration diagram is an introduction diagram that emphasizes the structural organization of the objects that send and receive messages. Graphically a collaboration diagram is a collection of vertices and arcs.

CLASS DIAGRAM:

            Class is nothing but a structure that contains both variables and methods.  The Class Diagram shows a set of classes, interfaces, and collaborations and their relating ships.  There is most common diagram in modeling the object oriented systems and are used to give the static view of a system.  It shows the dependency between the classes that can be used in our system.

The interactions between the modules or classes of our projects are shown below.  Each block contains Class Name, Variables and Methods.

CLASS:

   A description of set of objects that share the same attributes, operations, relationships, and semantics

State Chart Diagram

DATA FLOW DIAGRAMS: - VEHICLE SERVICE MANAGEMENT SYSTEM PROJECT

The DFD takes an input-process-output view of a system i.e. data objects flow into the software, are transformed by processing elements, and resultant data objects flow out of the software.  
Data objects represented by labeled arrows and transformation are represented by circles also called as bubbles. DFD is presented in a hierarchical fashion i.e. the first data flow model represents the system as a whole. Subsequent DFD refine the context diagram (level 0 DFD), providing increasing details with each subsequent level.  
The DFD enables the software engineer to develop models of the information domain & functional domain at the same time. As the DFD is refined into greater levels of details, the analyst perform an implicit functional decomposition of the system. At the same time, the DFD refinement results in a corresponding refinement of the data as it moves through the process that embody the applications.  
A context-level DFD for the system the primary external entities produce information for use by the system and consume information generated by the system. The labeled arrow represents data objects or object hierarchy.

RULES FOR DFD:

·         Fix the scope of the system by means of context diagrams.

·         Organize the DFD so that the main sequence of the actions

·         Reads left to right and top to bottom.

·          Identify all inputs and outputs.

·         Identify and label each process internal to the system with Rounded   circles.

·         A process is required for all the data transformation and Transfers. Therefore, never connect a data store to a data Source or the destinations or another data store with just a Data flow arrow.

·         Do not indicate hardware and ignore control information.

·         Make sure the names of the processes accurately convey everything the process is done.

·         There must not be unnamed process.

·         Indicate external sources and destinations of the data, with        Squares.

·         Number each occurrence of repeated external entities.

·         Identify all data flows for each process step, except simple Record retrievals.

·         Label data flow on each arrow.

·         Use details flow on each arrow.

·         Use the details flow arrow to indicate data movements.

E-R Diagrams: VEHICLE SERVICE MANAGEMENT SYSTEM PROJECT

    The Entity-Relationship (ER) model was originally proposed by Peter in 1976 [Chen76] as a way to unify the network and relational database views. Simply stated the ER model is a conceptual data model that views the real world as entities and relationships. A basic component of the model is the Entity-Relationship diagram which is used to visually represents data objects. Since Chen wrote his paper the model has been extended and today it is commonly used for database design For the database designer, the utility of the ER model is:

it maps well to the relational model. The constructs used in the ER model can easily be transformed into relational tables.

it is simple and easy to understand with a minimum of training. Therefore, the model can be used by the database designer to communicate the design to the end user.

In addition, the model can be used as a design plan by the database developer to implement a data model in a specific database management software.

CONNECTIVITY AND CARDINALITY

The basic types of connectivity for relations are: one-to-one, one-to-many, and many-to-many. A *one-to-one*(1:1) relationship is when at most one instance of a entity A is associated with one instance of entity B. For example, "employees in the company are each assigned their own office. For each employee there exists a unique office and for each office there exists a unique employee.

A *one-to-many*(1:N) relationships is when for one instance of entity A, there are zero, one, or many instances of entity B, but for one instance of entity B, there is only one instance of entity A. An example of a 1:N relationships is

a department has many employees

each employee is assigned to one department

A *many-to-many*(M:N) relationship, sometimes called non-specific, is when for one instance of entity A, there are zero, one, or many instances of entity B and for one instance of entity B there are zero, one, or many instances of entity A. The connectivity of a relationship describes the mapping of associated

ER NOTATION

           There is no standard for representing data objects in ER diagrams. Each modeling methodology uses its own notation. The original notation used by Chen is widely used in academics texts and journals but rarely seen in either CASE tools or publications by non-academics. Today, there are a number of notations used, among the more common are Bachman, crow's foot, and IDEFIX.

        All notational styles represent entities as rectangular boxes and relationships as lines connecting boxes. Each style uses a special set of symbols to represent the cardinality of a connection. The notation used in this document is from Martin. The symbols used for the basic ER constructs are:

entities are represented by labeled rectangles. The label is the name of the entity. Entity names should be singular nouns.

relationships are represented by a solid line connecting two entities. The name of the relationship is written above the line. Relationship names should be verbs

attributes, when included, are listed inside the entity rectangle. Attributes which are identifiers are underlined. Attribute names should be singular nouns.

cardinality of many is represented by a line ending in a crow's foot. If the crow's foot is omitted, the cardinality is one.

existence is represented by placing a circle or a perpendicular bar on the line. Mandatory existence is shown by the bar (looks like a 1) next to the entity for an instance is required. Optional existence is shown by placing a circle next to the entity that is optional

This project contains 3 modules namely :-

Ø  User Registration,

Ø  Service Charges,

Ø  Submission module,

User Creation:-

                   In this module  we are Registration  the  username ,password ,phno  and  vehicle no ,warranty and branch no’ are  user creations.

Service Charges:-

               In this module we are Registered the Customer Name, Vehicle no, Parts  and Servicing Charges as well as Paying the money.

Submission Module;-

                      In this module the dealer is handover to the Customer and before registering the  Customername,Vehicleno and phone.

Maintenance Order Registration and Warranties

ABOUT INTERNET AND INTRANET

Technologically, the Internet is network of computers. Not just a few special Computers, but over nine million of all kinds of computers. Similarly it is not just a network, but a network of networks hence the name and using TCP/IP (transmission control protocol and internet protocol).  
Internet is the name for a vast, worldwide system consisting of people, information and computers. Internet is global communication system of diverse, INTER connected computer NETWORK for the exchange of information of virtually every conceivable topic known to man.  
Internet is not just one thing. It is a lot of things to lot of people. In today’s world it is one of the most important commodity of life. The Internet is more important in what it enables than what it is, more of a phenomenon than fact.  
  
Intranet  
The classical definition of Intranet is the application of the Internet technologies to the internal business applications media most refer to the Intranet in terms of applying web technologies to information systems in the organization.

Introduction to HTML

    The hypertext markup language (HTML) is a simple markup language. Used to create a hypertext documents that are portable from one platform to another HTML documents are SGML (Standard generalized mark up language) documents with generic semantics that are appropriate for representing information from a wide range of applications. This specification defines HTML version 3.2. HTML 3.2 aims to capture recommended practice as of early ’96 and as such a replacement for HTML2.0 (RFC 1866).

            A set of instructions embedded in a document is called mark up language. These instructions describe what the document text means and how it should look like in a display. Hyper Text Mark Up language (HTML) is the language used to encode World Wide Web documents.

WHY TO USE HTML?

            Website is a collection of pages, publications, and documents that reside on web server. While these pages publications and a document as a formatted in a single format, you should use HTML for home page and all primary pages in the site. This will enable the millions of web users can easily access and to take advantage of your website.

HTML is considered first for formatting any new material you plan to publish on the web. HTML documents are platform independent, meaning that they don’t confirm to any standard. If they are created properly you can move home page to any server platform or you can access them with any complaint www browser.

STRUCTURE OF HTML

HTML elements perform a defined task. HTML uses two types of elements

. Empty Tags

            . Container Tags

            These tags differ because of what they represent. Empty tags represent formatting constricts such as line breaks and horizontal rules. Container tags define a section of text, formats and dot all of the selected text. A container tag has both a beginning and an ending.

METHOD ATTRIBUTE:

         The other required attribute for the

Top of Form

tag sets the methods by which the browser form’s data to the server for processing. There are two ways: the POST method and GET method. With POST method, the browser sends the data in two steps: the browser first contacts the form-processing server specified in the action attributes, and once contact is made, sends the data.

Bottom of Form

            The GET method in the other hand, contacts the form processing server and sends the form data in a single transaction step: the browser appends the data to the form’s action URL, separated by the question mark (?) character.

3.1 Front End Technology - Microsoft .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.

3.2   BACK END TECHNOLOGY: About Microsoft SQL Server 2000

Microsoft SQL Server is a Structured Query Language (SQL) based, client/server relational database. Each of these terms describes a fundamental part of the architecture of SQL Server.

Database

A database is similar to a data file in that it is a storage place for data. Like a data file, a database does not present information directly to a user; the user runs an application that accesses data from the database and presents it to the user in an understandable format.

A database typically has two components: the files holding the physical database and the database management system (DBMS) software that applications use to access data. The DBMS is responsible for enforcing the database structure, including:

·                     Maintaining the relationships between data in the database.

·                     Ensuring that data is stored correctly and that the rules defining data relationships are not violated.

·                     Recovering all data to a point of known consistency in case of system failures.

TESTING -VEHICLE SERVICE MANAGEMENT SYSTEM PROJECT

Testing is a process of executing a program with the indent of finding an error. Testing is a crucial element of software quality assurance and presents ultimate review of specification, design and coding.  
  
System Testing is an important phase. Testing represents an interesting anomaly for the software. Thus a series of testing are performed for the proposed system before the system is ready for user acceptance testing.  
  
A good test case is one that has a high probability of finding an as undiscovered error. A successful test is one that uncovers an as undiscovered error.

Testing Objectives:

1. Testing is a process of executing a program with the intent of finding an error  
  
2. A good test case is one that has a probability of finding an as yet undiscovered error  
  
3. A successful test is one that uncovers an undiscovered error  
  
Testing Principles:  
  
· All tests should be traceable to end user requirements  
  
· Tests should be planned long before testing begins  
  
· Testing should begin on a small scale and progress towards testing in large  
  
· Exhaustive testing is not possible  
  
· To be most effective testing should be conducted by a independent third party  
  
The primary objective for test case design is to derive a set of tests that has the highest livelihood for uncovering defects in software. To accomplish this objective two different categories of test case design techniques are used. They are  
  
White box testing.  
  
Black box testing.

White-box testing:

White box testing focus on the program control structure. Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.

Block-box testing:  
  
Black box testing is designed to validate functional requirements without regard to the internal workings of a program. Black box testing mainly focuses on the information domain of the software, deriving test cases by partitioning input and output in a manner that provides through test coverage. Incorrect and missing functions, interface errors, errors in data structures, error in functional logic are the errors falling in this category.

Testing strategies:

A strategy for software testing must accommodate low-level tests that are necessary to verify that all small source code segment has been correctly implemented as well as high-level tests that validate major system functions against customer requirements.  
  
Testing fundamentals:  
  
Testing is a process of executing program with the intent of finding error. A good test case is one that has high probability of finding an undiscovered error. If testing is conducted successfully it uncovers the errors in the software. Testing cannot show the absence of defects, it can only show that software defects present.  
  
Testing Information flow:  
  
Information flow for testing flows the pattern. Two class of input provided to test the process. The software configuration includes a software requirements specification, a design specification and source code.  
  
Test configuration includes test plan and test cases and test tools. Tests are conducted and all the results are evaluated. That is test results are compared with expected results. When erroneous data are uncovered, an error is implied and debugging commences.  
  
Unit testing:  
  
Unit testing is essential for the verification of the code produced during the coding phase and hence the goal is to test the internal logic of the modules. Using the detailed design description as a guide, important paths are tested to uncover errors with in the boundary of the modules. These tests were carried out during the programming stage itself. All units of ViennaSQL were successfully tested.  
  
Integration testing :  
  
Integration testing focuses on unit tested modules and build the program structure that is dictated by the design phase.  
  
System testing:  
  
System testing tests the integration of each module in the system. It also tests to find discrepancies between the system and it’s original objective, current specification and system documentation. The primary concern is the compatibility of individual modules. Entire system is working properly or not will be tested here, and specified path ODBC connection will correct or not, and giving output or not are tested here these verifications and validations are done by giving input values to the system and by comparing with expected output. Top-down testing implementing here.  
  
Acceptance Testing:  
  
This testing is done to verify the readiness of the system for the implementation. Acceptance testing begins when the system is complete. Its purpose is to provide the end user with the confidence that the system is ready for use. It involves planning and execution of functional tests, performance tests and stress tests in order to demonstrate that the implemented system satisfies its requirements.  
  
Tools to special importance during acceptance testing include:  
  
Test coverage Analyzer – records the control paths followed for each test case.  
  
Timing Analyzer – also called a profiler, reports the time spent in various regions of the code are areas to concentrate on to improve system performance.  
  
Coding standards – static analyzers and standard checkers are used to inspect code for deviations from standards and guidelines.  
  
Test Cases:  
  
Test cases are derived to ensure that all statements in the program have been executed at least once during testing and that all logical conditions have been executed.  
  
Using White-Box testing methods, the software engineer can drive test cases that  
  
· Guarantee that logical decisions on their true and false sides.  
  
· Exercise all logical decisions on their true and false sides.  
  
· Execute all loops at their boundaries and with in their operational bounds.  
  
· Exercise internal data structure to assure their validity.  
  
The test case specification for system testing has to be submitted for review before system testing commences.

CONCLUSION: VEHICLE SERVICE MANAGEMENT SYSTEM PROJECT

The package was designed in such a way that future modifications can be done easily. The following conclusions can be deduced from the development of the project.  
  
  
Automation of the entire system improves the efficiency  
  
It provides a friendly graphical user interface which proves to be better when compared to the existing system.  
  
It gives appropriate access to the authorized users depending on their permissions.  
  
It effectively overcomes the delay in communications.  
  
Updating of information becomes so easier.  
  
System security, data security and reliability are the striking features.  
  
The System has adequate scope for modification in future if it is necessary.