**1. ABSTRACT**

This project is entitled as “**Advanced** **Car pooling Application**” is developed as a mobile application using client server tools programs like Asp.net as front end, C# as coding language and SQL server as backend.

The main objective of this project is to reduce the travelling cost while travelling through private vehicles like car, cabs and etc from the same institute. In this fast life, individuals find difficult to interact with people due to lack of time even though they belong to same place as the destination. Still many people travelling alone in a car for long distance in the same institute. This makes more expensive for the traveler. According to the survey, group traveling will reduce the cost of travel, provides more secured, fuel efficiency, less pollutes the environment and etc. Group travel will give us more advantages than single travel.

In order to connect people, this application is developed to be connected with those people who are travelling to the same destination or via destination. Also one should contact the people in their circle directly if they want to travel together with them. This application contains two medium namely provider and seeker. Providers are the one who initiate the travel and fix the destination and seekers are the one who accepts to travel with the provider. This process made more secured, like user can search:

* Gender wise (More safe for ladies)
* Age wise (Won’t make travel bore)
* Department wise (Can discuss something interesting)
* Staff or student wise (Can make trip useful)

Here our application mainly concentrates on simplicity and security and how to travel efficiently using multiple references. Rating will be provided to the provider, if he provider is having less or worst rating in the history then seeker can avoid travel with them. After the completion of the travel, both provider and seeker can rate about their travel and about the person. This application also allows the users to provide the feedback about the drivers which be useful for the forthcoming users to select the drivers accordingly. Also cancellation of trip is available for both provider and seeker. According to the technology, client server technology has been implemented with centralized server and centralized data sharing.

**1.2 Modules**

* Provider
* Seeker
* Travel Confirmation
* Manage Details and comments
* Mobile application and server

**1.3 Module Description:**

**Provider:**

Sign up is the initial process of the project. Here the provider can create the free sign up with their personal details and create the travel details like from, via and to destination. After creating the destination the provider need to add their license details, vehicle type and number of members going to travel with him.

**Seeker**

The same process is for seeker. Seeker need to sign up with their personal details. Seeker can search for the travel destination. A multi option list box has been provided to seeker for selecting the source and destination also the date of the travel. Once the seeker selected the source and destination, they can see the provider’s basic details like name and car details. A request will be sent to the provider from the seeker. Now provider can see the seeker’s request.

**Travel Confirmation**

The travel confirmation will be done by the provider only. The provider can view the seeker’s full details after the confirmation only. After the confirmation the provider and seeker can view the full details. This option will be enabled after the confirmation only. Once the travel got confirmed the search details will disable for other search. So that provider will not disturbed from various unwanted requests.

**Manage Details and comments**

This module has more details like add, update, view and delete. In case the provider cant able to travel on the same day means, he can update the details. If seeker is willing they can travel in the other day. Else the provider can delete the posted travel details permanently. Commenting options available in this module. Using this option provider or seeker can view the corresponding comments of the user before the travel. Comments can be viewed only by selecting the users.

**Mobile application and server**

This is most important module in this project; this is because the updating details will be stored in the centralized server and the sub servers. A prior server will be allocated to access these details. They will be permitted to update the details too. All the centralized details will be available in the web server and mobile application so that user can able to access the server anywhere at any time. This will be more security provided to the centralized server and high bandwidth will be allocated to access more number of users at a same time. A dedicated mobile application is developed using this module.

**2. SYSTEM CONFIGURATION**

**2.1 HARDWARE SPECIFICATION**

PROCESSOR : Intel Pentium IV 1.8 GHz

MOTHERBOARD : Intel 915GVSR chipset board

RAM : 1 GB DDR2 RAM

HARD DISK DRIVE : 160 GB

FLOPPY DRIVE : 1.44 MB

DVD/CD DRIVE : Sony 52 x Dual layer drive

MONITOR : 17” Color TFT Monitor

KEYBOARD : Multimedia Keyboard 108 Keys

MOUSE : Logitech Optical Mouse

CABINET : ATX iball.

HUB : Compex 16 lines.

BANDWIDTH : 100 mbps.

**2.2 SOFTWARE CONFIGURATION**

FRONTEND : ASP.NET 2012

CODING LANDUAGE : C#

BACK END : SQL SERVER 2010

CLIENT SERVER TOOL : AJAX 2.0

OPERATING SYSTEMS : Microsoft Windows 7

DOCUMENTATION : Microsoft word 2007.

SCRIPTING LANGUAGE : Java Script

**3. SYSTEM STUDY**

**3.1 EXISTING SYSTEM STUDY**

According to the existing system, here is no user based car sharing methods. The system study phase studies the problem, identifies alternate solutions evaluate those solutions and finally recommends the best solution for the existing system. The system study gives the structure & function of the system. The system Study can be performed only on an existing system. The system study gives an idea of then user requirements. A detailed system study is an essential for developing an efficient system.

**OBSERVATION:**

Observation of the functioning of the existing system gives the Idea for the design of the new system. This will figure of the pitfalls of the Current system. It is helpful to understand and study the entire current system. By observation we can point out the changes needed to the existing system. It also validates the data gathered by other means. It also gives a better understanding of the workloads & pressures faced.

**Limitations of existing system:**

At present, the requirements of the system are done manually and hence there are number of limitations. The various activities are carried out by different section depending on their activity. The various status based information is maintained in books and files, which is very difficult.

* Data security is less.
* Calculations are done manually.
* No Tracking System Available
* No Delay Detail analysis

Since at present everything is done manually, it is having lot of drawbacks. The major drawback of the present system is the bulk amount of physical volume of the data making information search & retrieval a tedious process. There are changes of papers containing information. Because of all these problems of the present system development by the computerized the system, which could solve all these, was inevitable. In the case of manual entry, chances of entering incorrect data are high.

**Disadvantages of the existing system**

* Still in the current system is not available in any application for car sharing during long travels.
* Only manual process is available like, discussing through phones, SMS and mails.
* Only known persons can travel with their friends.
* No safety for girls.
* In case of travelling alone for a long distance means, the driver may get bored and he will face more expensive for the fuel.
* Unknown persons will not be allowed in their personal cars for long travels.
* Problems may occur in sharing the money and about the routes.
* No dedicated mobile application is available.

**3.2 PROPOSED SYSTEM:**

The proposed s system tries to avoid the problems raised in the existing system. The main automation process is the mobile based application. Here the application is developed in a web based method and hosted in the centralized server. The centralized server details can be accessed through the mobile application. This application is a freeware, so any user can download from the Google market. It automates the entire process of courier management System. It should have a provision to maintain master entries, provision to Prepare inward, delivery details etc. It should also provide often needed reports to satisfy the need of the concern. With the help of help messages & validation checks the new system can contribute a lot to the accuracy of data report generation is very easy. Computerizing the system can handle large inflow of data. It is fast flexible & user-friendly than the manual one.

**MERITS OF PROPOSED SYSTEM:**

* As the proposed system is system oriented it is faster than the manual process
* Absolutely Freeware
* Tracking made simple
* Stage wise tracking available
* It also minimizes the errors in the data entry.
* It improves customer satisfaction due to the improvement in the technology

In added with, this proposed system will make a major impact in within the travellers. Still many travellers are seeking for these kinds of applications. The proposed application is developed in both personal computer application as well as mobile application. And there are two major factors in this application they are:

* Provider
* Seeker

Provider is the person who creates the travel; all the travel details will be centralized in a web server. Seeker can search from and to location according to their travel date and time. The server will display the provider’s details. So that the seeker and provider can make their travel plans according to their wish.

**Advantages of the proposed system**

* A dedicated mobile application has been proposed for travellers, those who travel frequently.
* All the processes are automated in a single window, using client server technology. with
* Can travel with unknown person, safety guaranteed.
* Can travel more distance for less amount of money, can’t get bored during the travel.
* After knowing all the travel details, the phone number and email will be shared. So that phone number privacy has been implemented.
* Provider can cancel the travel plan at any time in case of any emergency.

**3.3 FEASIBILITY STUDY**

The feasibility of the project is analyzed in this phase and business proposal is put forth with a very general plan for the project and some cost estimates. During system analysis the feasibility study of the proposed system is to be carried out. This is to ensure that the proposed system is not a burden to the company. For feasibility analysis, some understanding of the major requirements for the system is essential.

Three key considerations involved in the feasibility analysis are

* ECONOMICAL FEASIBILITY
* TECHNICAL FEASIBILITY
* SOCIAL FEASIBILITY

**ECONOMICAL FEASIBILITY**

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

### TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

**SOCIAL FEASIBILITY**

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity.

The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

**3.4 Cost Estimating & Scheduling**

In developing cost estimates for a system several cost elements are considered.

1. Personnel Cost

2. Facility Cost

3. Operating Cost

**Personnel Cost:**

Personnel cost includes staff salaries and benefits as well as pay for those involved in developing the system.

**Facility cost:**

Facility cost is expenses incurred in the prepration of the physical site where application or computer will be in operation.

**Operating cost:**

Operating cost include all costs associated with the day to day operation of the system. This project is not costly. It includes only hardware cost for purchase of modem, hub and cable

**4.0 SYSTEM DESIGN**

**4.1 Data Flow Diagram**

**Level 0:**

Provider, Seeker

Database

Request for username & Password

Retrieve

Response

**Level 1:**

Provider

Username, password

Location from and to,date,time

Edit, update, delete

Update, delete

Sugg type, name

details

Login details

Travel

Info

Seeker

Contravel

Old password, new password

**Level 2:**

Seeker

Location,time

Name, description

Name, answer, date, time

Login details

Travel

suggestdet

Uname and psw

contravel

**4.2 ER diagram**

Provider

Create

Travel

Provider info

Manage

Seeker

Reply

Contravel

Confirm

View

View

**1**

**N**

**N**

**1**

**N**

**1**

**N**

**N**

**N**

**N**

**4.3 Table Design**

**Table name: Login**

**Primary Key: username**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fields** | **Datatype** | **Size** | **Constraints** | **Description** |
| Username | varchar | 30 | Primary Key | username |
| Password | varchar | 50 | Not null | password |

**Table name : User Registration**

**Primary Key:** id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fields** | **Datatype** | **Size** | **Constraints** | **Description** |
| uid | Int | 30 | Primary Key | User id number |
| Uname | Varchar | 20 | Foreign key | User Name |
| Pass | Varchar | 30 | Not null | User Password |
| gender | Varchar | 10 | Not null | User gender |
| occ | Varchar | 20 | Not null | Occupation |
| Type | Varchar | 10 | Not null | Type of business |
| DOB | Date | 20 | Not null | Date of birth |
| Street1 | Varchar | 150 | Not null | address1 |
| Street2 | Varchar | 150 | Not null | address2 |
| City | Varchar | 20 | Not null | City name |
| State | Varchar | 20 | Not null | State |
| Pin | Varchar | 20 | Not null | Pincode |
| Phoneno | Int | 32 | Not null | phone number |
| email | Varchar | 20 | Not null | email id |

**Table name :Vehical registration**

**Primary Key:**id

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fields** | **Datatype** | **Size** | **Constraints** | **Description** |
| tid | varchar | 40 | Primary Key | Travel identification number |
| Vehical no | varchar | 32 | Not null | Vehicle number |
| Vehical name | Varchar | 20 | Not Null | Name of vehicle |
| Vehical type | Varchar | 32 | Not Null | Type of vehicle |
| From | Varchar | 32 | Not Null | Starting point |
| to | Varchar | 32 | Not Null | Destination point |
| Via | Varchar | 32 | Not Null | Via route |
| Date | Varchar | 32 | Not Null | Journey date |
| Time | Varchar | 32 | Not Null | Starting time |

**Table name : confirmation**

**Primary Key: id**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fields** | **Datatype** | **Size** | **Constraints** | **Description** |
| suid | Varchar | 40 | Primary Key | Seeker User identification |
| Sname | Varchar | 20 | Foreign key | Seeker name |
| gender | Varchar | 10 | Not null | User gender |
| occ | Varchar | 20 | Not null | occupation |
| Type | Varchar | 10 | Not null | Type of business |
| DOB | Date | 20 | Not null | Date of birth |
| Street1 | Varchar | 150 | Not null | address1 |
| Street2 | Varchar | 150 | Not null | address2 |
| City | Varchar | 20 | Not null | City name |
| Confirm | Varchar | 20 | Not null | Seeker and provide confirmation |

**Table name : Rating**

**Primary Key: paymentid**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Fields** | **Datatype** | **Size** | **Constraints** | **Description** |
| Uid | Varchar | 40 | Primary Key | Provider user identification |
| Suid | Varchar | 20 | Not Null | Seeker User identification |
| Star | Varchar | 10 | Not Null | Star rating |
| Comment | Varchar | 20 | Not Null | Comments |

**5.0 SOFTWARE SPECIFICATION**

**5.1 ABOUT FRONT END**

**ASP.NET**

The .NET Framework (pronounced dot net) is a software framework developed by Microsoft that runs primarily on Microsoft Windows. It includes a large library and provides language interoperability (each language can use code written in other languages) across several programming languages. Programs written for the .NET Framework execute in a software environment (as contrasted to hardware environment), known as the Common Language Runtime (CLR), an application virtual machine that provides services such as security, memory management, and exception handling. The class library and the CLR together constitute the .NET Framework.

The .NET Framework's Base Class Library provides user interface, data access, database connectivity, cryptography, web application development, numeric algorithms, and network communications. Programmers produce software by combining their own source code with the .NET Framework and other libraries. The .NET Framework is intended to be used by most new applications created for the Windows platform. Microsoft also produces an integrated development environment largely for .NET software called Visual Studio.

**COMMON LANGUAGE RUNTIME**

The Common Language Runtime (CLR) is the virtual machine component of Microsoft's .NET framework and is responsible for managing the execution of .NET programs. In a process known as just-in-time compilation, the compiled code is converted into machine instructions that, in turn, are executed by the computer's CPU. The CLR provides additional services including memory management, type safety and exception handling. All programs written for the .NET framework, regardless of programming language, are executed by the CLR. It provides exception handling, garbage collection and thread management. CLR is common to all versions of the .NET framework.

**Garbage collection**

Garbage Collection is the concept of the development environment disposes of resources that are using up memory within the application that are no longer actually in use. You can think of resources that are taking up memory and aren't being used or accessed as "garbage" and the act of garbage collection is actually cleaning and disposing of these things.

Nearly everything that you do within an application is going to take up memory, which will be allocated and accessed at specific times throughout the lifespan of the execution of the program. However, if you never dispose of any of the memory and resources, your application will just continue to take up more and more of your system resources .

The .NET Framework's garbage collector manages the allocation and release of memory for your application. Each time you create a new object, the common language runtime allocates memory for the object from the managed heap. As long as address space is available in the managed heap, the runtime continues to allocate space for new objects. However, memory is not infinite. Eventually the garbage collector must perform a collection in order to free some memory. The garbage collector's optimizing engine determines the best time to perform a collection, based upon the allocations being made. When the garbage collector performs a collection, it checks for objects in the managed heap that are no longer being used by the application and performs the necessary operations to reclaim their memory.

**Common Language Specification**

A CLS is a document that says how computer programs can be turned into bytecode. When several languages use the same bytecode, different parts of a program can be written in different languages. Microsoft uses a Common language specification for their dotnet Framework. It was always a dream of Microsoft to unite all different languages in to one umbrella and CLS is one step towards that. Microsoft has defined CLS which are nothing but guidelines that language to follow so that it can communicate with other .NET languages in a seamless manner.

The Common Language Specification is a set of constructs and constraints that serves as a guide for library writers and compiler writers. It allows libraries to be fully usable from any language supporting the CLS, and for those languages to integrate with each other. The Common Language Specification is a subset of the common type system. The Common Language Specification is also important to application developers who are writing code that will be used by other developers. When developers design publicly accessible APIs following the rules of the CLS, those APIs are easily used from all other programming languages that target the common language runtime.

**COMMON INFRASTRUCTURE LIBRARIES FOR .NET:**

The first of these libraries is a portable logging abstraction, Common Logging. Other libraries are planned but there is not a defined roadmap at the moment. Suggestions have ranged from a validation library to utilities for thread local storage.

There are a variety of logging implementations for .NET currently in use, log4net, Enterprise Library Logging, NLog, to name the most popular. The downside of having different implementation is that they do not share a common interface and therefore impose a particular logging implementation on the users of your library.

Common Logging library introduces a simple abstraction to allow you to select a specific logging implementation at runtime. Thus you can defer the decision what particular logging library to use until deployment. Adapters are used for plugging a particular logging system into Common Logging.

The library is based on work done by the developers of IBatis.NET and it's usage is inspired by log4net. Many thanks to the developers of those projects! The library is available with both debug and strongly signed release assemblies.

**Memory Management in .NET**

In .NET memory is managed through the use of Managed Heaps. Generally in case of other languages, memory is managed through the Operating System directly. The program is allocated with some specific amount of memory for its use from the Raw memory allocated by the Operating system and then used up by the program. In case of .NET environment, the memory is managed through the CLR (Common Language Runtime) directly and hence we call .NET memory management as Managed Memory Management.

**Allocation of Memory**

Generally .NET is hosted using Host process, during debugging .NET creates a process using VSHost.exe which gives the programmer the basic debugging facilities of the IDE and also direct managed memory management of the CLR. After deploying your application, the CLR creates the process in the name of its executable and allocates memory directly through Managed Heaps.

When CLR is loaded, generally two managed heaps are allocated; one is for small objects and other for Large Objects. We generally call it as SOH (Small Object Heap) and LOH (Large Object Heap). Now when any process requests for memory, it transfers the request to CLR, it then assigns memory from these Managed Heaps based on their size. Generally, SOH is assigned for the memory request when size of the memory is less than 83 KBs( 85,000 bytes). If it is greater than this, it allocates memory from LOH. On more and more requests of memory .NET commits memory in smaller chunks.

Now let’s come to processes. Generally a process can invoke multiple threads, as multi-threading is supported in .NET directly. Now when a process creates a new thread, it creates its own stack, i.e. for the main thread .NET creates a new Stack which keeps track of all information associated with that particular thread. It keeps information regarding the current state of the thread, number of nested calls etc. But every thread is using the same Heap for memory. That means, Heaps are shared through all threads.

Upon request of memory from a thread say, .NET allocates its memory from the shared Heap and moves its pointer to the next address location. This is in contrast to all other programming languages like C++ in which memory is allocated in linked lists directly managed by the Operating system, and each time memory requests is made by a process, Operating system searches for the big enough block. Still .NET win32 application has the limitation of maximum 2GB memory allocation for a single process.

32 bit processors have 32 bits of address space for locating a single byte of data. This means each 2^32 unique address locations that each byte of data can locate to, means 4.2 billion unique addresses (4GB). This 4GB memory is evenly distributed into two parts, 2 GB for Kernel and 2 GB for application usage.

**PERFORMANCE AND SCALABILITY**

This guide provides end-to-end guidance for managing performance and scalability throughout your application life cycle to reduce risk and lower total cost of ownership. It provides a framework that organizes performance into a handful of prioritized categories where your choices heavily impact performance and scalability success. The logical units of the framework help integrate performance throughout your application life cycle. Information is segmented by roles, including architects, developers, testers, and administrators, to make it more relevant and actionable. This guide provides processes and actionable steps for modeling performance, measuring, testing, and tuning your applications. Expert guidance is also provided for improving the performance of managed code, ASP.NET, Enterprise Services, Web services, remoting, ADO.NET, XML, and SQL Server.

**.NET SECURITY**

The Microsoft .NET Framework gives numerous techniques and a vast range of types in the security namespaces to help you build secure code and create secure Web applications. This chapter defines the .NET Framework security landscape by briefly introducing the security benefits of managed code development. This chapter also introduces and contrasts the two complimentary forms of security that are available to .NET Framework applications: user security and code security. Finally, the chapter briefly examines the security namespaces that you use to program .NET Framework security.

This chapter emphasizes how .NET Framework security applies to ASP.NET Web applications and Web services.This chapter describes the security benefits inherent in using the .NET Framework and explains the complementary features of .NET Framework user (or role-based) security and .NET Framework code-based (or code access) security. We recommend that you use this chapter as follows:

• Understand the two-layered defense provided by the .NET Framework. Role-based security allows you to control user access to application resources and operations, while code access security can control which code can access resources and perform privileged operations.

• Create applications that use the security concepts in this chapter. This chapter tells you when you should use user-based security and when you should use code-based security. After reading this chapter, you will be able to identify how any new applications you create can be more secure by using role-based or code-based security.

**Managed Code Benefits**

Developing .NET Framework applications provides you with some immediate security benefits, although there are still many issues for you to think about. These issues are discussed in the Building chapters in Part III of this guide.

.NET Framework assemblies are built with managed code. Compilers for languages, such as the Microsoft Visual C# development tool and Microsoft Visual Basic .NET development system, output Microsoft intermediate language (MSIL) instructions, which are contained in standard Microsoft Windows portable executable (PE) .dll or .exe files. When the assembly is loaded and a method is called, the method's MSIL code is compiled by a just-in-time (JIT) compiler into native machine instructions, which are subsequently executed. Methods that are never called are not JIT-compiled.

The use of an intermediate language coupled with the run-time environment provided by the common language runtime offers assembly developers immediate security advantages.

• File format and metadata validation. The common language runtime verifies that the PE file format is valid and that addresses do not point outside of the PE file. This helps provide assembly isolation. The common language runtime also validates the integrity of the metadata that is contained in the assembly.

• Code verification. The MSIL code is verified for type safety at JIT compile time. This is a major plus from a security perspective because the verification process can prevent bad pointer manipulation, validate type conversions, check array bounds, and so on. This virtually eliminates buffer overflow vulnerabilities in managed code, although you still need to carefully inspect any code that calls unmanaged application programming interfaces (APIs) for the possibility of buffer overflow.

• Integrity checking. The integrity of strong named assemblies is verified using a digital signature to ensure that the assembly has not been altered in any way since it was built and signed. This means that attackers cannot alter your code in any way by directly manipulating the MSIL instructions.

• Code access security. The virtual execution environment provided by the common language runtime allows additional security checks to be performed at runtime. Specifically, code access security can make various run-time security decisions based on the identity of the calling code.

**User vs. Code Security**

User security and code security are two complementary forms of security that are available to .NET Framework applications. User security answers the questions, "Who is the user and what can the user do?" while code security answers the questions "Where is the code from, who wrote the code, and what can the code do?" Code security involves authorizing the application's (not the user's) access to system-level resources, including the file system, registry, network, directory services, and databases. In this case, it does not matter who the end user is, or which user account runs the code, but it does matter what the code is and is not allowed to do.

The .NET Framework user security implementation is called role-based security. The code security implementation is called code access security.

**Role-Based Security**

.NET Framework role-based security allows a Web application to make security decisions based on the identity or role membership of the user that interacts with the application. If your application uses Windows authentication, then a role is a Windows group. If your application uses other forms of authentication, then a role is application-defined and user and role details are usually maintained in SQL Server or user stores based on Active Directory.

The identity of the authenticated user and its associated role membership is made available to Web applications through Principal objects, which are attached to the current Web request.

**Code Access Security**

Code access security authorizes code when it attempts to access secured resources, such as the file system, registry, network, and so on, or when it attempts to perform other privileged operations, such as calling unmanaged code or using reflection.

Code access security is an important additional defense mechanism that you can use to provide constraints on a piece of code. An administrator can configure code access security policy to restrict the resource types that code can access and the other privileged operations it can perform. From a Web application standpoint, this means that in the event of a compromised process where an attacker takes control of a Web application process or injects code to run inside the process, the additional constraints that code access security provides can limit the damage that can be done.

A logical view of how code access security is used in a Web application to constrain the application's access to system resources, resources owned by other applications, and privileged operations, such as calling unmanaged code.

The authentication (identification) of code is based on evidence about the code, for example, its strong name, publisher, or installation directory. Authorization is based on the code access permissions granted to code by security policy. For more information about .NET Framework code access security.

**.NET Framework Runtime Bundling and Installation**

The .NET Framework has been bundled with Windows for the last six years. Windows Vista first included .NET pre-installed (version 3.0 of the framework). Windows 7 includes version 3.5. Windows 8 includes version 4.5/4.0 but requires a download for version 2.0/3.0/3.5 and drops support entirely for 1.0/1.1. For earlier versions of Windows, the .NET runtime must be downloaded and installed on each PC by a user logged on with administrative privileges. The download size ranges from 50MB to up to 200MB depending on configuration and version. The .NET Framework is made available for direct download from Microsoft's website as well as Windows Update where it is listed as an optional update, meaning that it will not be installed by default unless patching an already-installed version (see screenshot). Most users only apply critical/high priority updates as these are the only ones Windows is configured to automatically apply. The installation process can take 15 minutes or more and requires closing all running programs and restarting the PC, so even in a situation with admin rights, installing and then removing is unreasonable.

**.NET Portability**

Several people have also asked about whether .NET itself can be made portable. The answer is no. NET is heavily tied to the Windows operating system and requires an administrator to install. Plus, it is a closed source, commercial product, so, legally, we couldn't do it, anyway. So, we're left with just analyzing what PCs already have it and what PCs can have it added.

**.NET Framework Availability By Version and Operating System**

Using the compatibility information by operating system, compatibility of .NET versions with each other, bundling information with operating system and some quick overview statistics of operating system usage online, I've compiled a quick chart that shows you where .NET can be expected, where it's available for download and where it simply won't work.

**SCALABILITY**

Scalability is the capability to increase resources to yield a linear (ideally) increase in service capacity. The key characteristic of a scalable application is that additional load only requires additional resources rather than extensive modification of the application itself.

Although raw performance makes a difference in determining the number of users that an application can support, scalability and performance are two separate entities. In fact, performance efforts can sometimes be opposed to scalability efforts.

**5.2 ABOUT BACK END**

**SQL SERVER:**

**Databases**

A database in Microsoft SQL Server consists of a collection of tables that contain data and other objects, such as views, indexes, stored procedures, and triggers, defined to support activities performed with the data. The data stored in a database is usually related to a particular subject or process, such as inventory information for a manufacturing warehouse.

SQL Server can support many databases. Each database can store either interrelated or unrelated data from other databases. For example, a server can have one database that stores personnel data and another that stores product-related data. Alternatively, one database can store current customer order data, and another related database can store historical customer orders used for yearly reporting.

Before you create a database, it is important to understand the parts of a database and how to design these parts to ensure that the database performs well after it is implemented. It is recommended that you do not create any user objects, such as tables, views, stored procedures, or triggers, in the master database. The master database contains the system tables that store the system information used by SQL Server, such as configuration option settings.

**Stored Procedures**

When you create an application with Microsoft SQL Server, the Transact-SQL programming language is the primary programming interface between your applications and the SQL Server database. When you use Transact-SQL programs, two methods are available for storing and executing the programs. You can store the programs locally and create applications that send the commands to SQL Server and process the results, or you can store the programs as stored procedures in SQL Server and create applications that execute the stored procedures and process the results.

Stored procedures in SQL Server are similar to procedures in other programming languages in that they can:

Accept input parameters and return multiple values in the form of output parameters to the calling procedure or batch.

Contain programming statements that perform operations in the database, including calling other procedures.

Return a status value to a calling procedure or batch to indicate success or failure (and the reason for failure).

You can use the Transact-SQL EXECUTE statement to run a stored procedure. Stored procedures are different from functions in that they do not return values in place of their names and they cannot be used directly in an expression. The benefits of using stored procedures in SQL Server rather than Transact-SQL programs stored locally on client computers are:

**They allow modular programming.**

You can create the procedure once, store it in the database, and call it any number of times in your program. Stored procedures can be created by a person who specializes in database programming, and they can be modified independently of the program source code.

**They allow faster execution.**

If the operation requires a large amount of Transact-SQL code or is performed repetitively, stored procedures can be faster than batches of Transact-SQL code. They are parsed and optimized when they are first executed, and a compiled version of the stored procedure remains in memory cache for later use. This means the stored procedure does not need to be reparsed and reoptimized with each use resulting in much faster execution times.

**They can reduce network traffic.**

An operation requiring hundreds of lines of Transact-SQL code can be performed through a single statement that executes the code in a procedure, rather than by sending hundreds of lines of code over the network.

**They can be used as a security mechanism.**

Users can be granted permission to execute a stored procedure even if they do not have permission to execute the procedure's statements directly.

A SQL Server stored procedure is created with the Transact-SQL CREATE PROCEDURE statement and can be modified with the ALTER PROCEDURE statement. The stored procedure definition contains two primary components: the specification of the procedure name and its parameters, and the body of the procedure, which contains Transact-SQL statements that perform the procedure's operations.

**Optimizing Database Performance**

The goal of optimizing database performance is to minimize the response time for each query and to maximize the throughput of the entire database server by minimizing network traffic, disk I/O, and CPU time. Understanding how to design the logical and physical structure of the data, tune queries, and configure Microsoft SQL Server and the operating system can help optimize database performance.

**Replication**

Microsoft SQL Server replication is the process of copying and distributing data and database objects from one database to another and then synchronizing between databases for consistency.

Using replication, you can distribute data to different locations, to remote or mobile users over a local area network, using a dial-up connection, and over the Internet. Replication also allows you to enhance application performance, physically separate data based on how it is used (for example, to separate online transaction processing (OLTP) and decision support systems), or distribute database processing across multiple servers.

**Replication Types (Enterprise Manager)**

Microsoft SQL Server provides the following types of replication that you can use in your distributed applications:

* Snapshot replication
* Transactional replication
* Merge replication

Each type provides different capabilities depending on your application and different levels of ACID properties of transactions and site autonomy. For example, merge replication allows users to work and update data autonomously, although ACID properties are not assured. Instead, when servers are reconnected, all sites in the replication topology converge to the same data values. Transactional replication maintains transactional consistency, but Subscriber sites are not as autonomous as they are in merge replication because Publishers and Subscribers generally must be connected reliably and continuously for updates to be propagated to Subscribers.

It is common for the same application to use multiple replication types and options. Some of the data in the application may not require any updates at Subscribers, some sets of data may require updates infrequently, with updates made at only one or a few servers, while other sets of data may need to be updated daily at multiple servers.

Which type of replication you choose for your application depends on your requirements based on distributed data factors, whether or not data will need to be updated at the Subscriber, your replication environment, and the needs and requirements of the data that will be replicated. For more information, see Planning for Replication.

Each type of replication begins with generating and applying the snapshot at the Subscriber, so it is important to understand snapshot replication in addition to any other type of replication and options you choose.

# SQL Connection

SQL Connection is an application program interface (API) that enables Synergy applications to use SQL‑based functions to access and manipulate data from various database systems. It consists of two types of functions that are based on standard SQL‑based operations:

•The database functions are directly related to SQL‑based operations and data access. They greatly simplify application development by reducing the total number of calls needed to accomplish a wide variety of SQL functions. See [“Database Functions.”](http://docs.synergyde.com/sql/3_DB_Func.html#Rsaf21898)

•The utility functions enable you to get information, map error codes, and set date and time options during the execution of your Synergy application. See [“Utility Functions.”](http://docs.synergyde.com/sql/4_Utl_Func.html#Rsag35374)

Each SQL Connection function generates a value and can be used any place a literal can be used in a Synergy program. Except where noted, SQL Connection functions work with both traditional Synergy and Synergy .NET.

There is a glossary to clarify some of the terms we use, but throughout we assume you are familiar with relational database management system (RDBMS) and SQL concepts. For information on a specific database operation, you may need to refer to the documentation for your database.

## Features and supported databases

SQL Connection conforms to ANSI‑standard database communication methods SQLCA and SQLDA (ANSI 89) and supports

* MySQL (version 5.1 and higher) on supported Windows, Linux, and AIX platforms.
* Oracle (version 10 and higher) on all supported Windows, UNIX, and OpenVMS platforms. Use version 10.2.0.4 or higher for OCI‑related fixes.
* SQL Server (version 2008 and higher) on supported Windows platforms. Note that for SQL Server 2008, only the VTX12\_SQLNATIVE driver is supported.
* Synergy DBMS (version 7.1 and higher) on all supported Windows, UNIX, and OpenVMS platforms.

SQL Connection has limited support for the following database systems. Note that support for these databases may require assistance from Synergex Professional Services and additional support fees. Contact your Synergy/DE account manager for details. (See [“Product support information”](http://docs.synergyde.com/sql/Preface_sql.html#Rsac32261).)

* Informix on UNIX systems
* ODBC‑compliant databases on Windows systems
* Oracle Rdb on OpenVMS systems
* Sybase on Windows and UNIX systems.Additional SQL Connection features include record‑oriented or column‑oriented bulk data access.
* built‑in statement caching and SQL prefetch caching.
* automatic data‑type conversion and binding of Synergy DBL variables with data.
* independence from Synergy DBL file I/O operations. support for up to seven concurrent database connections per program on UNIX and OpenVMS, and up to 100 concurrent connections per program on Windows (for multi‑threading). You can open multiple databases simultaneously, and you can open the same database more than once in an application (under the same user name or under a different user name).

**DATAREADER:**

In ADO.NET, a DataReader is a broad category of objects used to sequentially read data from a data source. DataReaders provide a very efficient way to access data, and can be thought of as a Firehose cursor from ASP Classic, except that no server-side cursor is used. A DataReader parses a Tabular Data Stream from Microsoft SQL Server, and other methods of retrieving data from other sources.

A DataReader is usually accompanied by a Command object that contains the query, optionally any parameters, and the connection object to run the query on.

**Strong vs Weak Typing**

When using a DataReader to retrieve data, the developer can choose to read field values in strongly typed manner or a weakly typed manner, returning then as System. Objects . Both approaches have their pros and cons.

Using the strongly typed retrieval methods can be more cumbersome, especially without specific knowledge of the underlying data. Numeric values in the database can translate to several .NET types: Int16, Int32, Int64, Float, Decimal, or Currency. Trying to retrieve a value using the wrong type results in an exception being thrown, which stops code from running further, and slows the application down. This is also true when you use the right type, but encounter a DbNull value ( this can be avoided by using the IsDbNull boolean function of the DataReader class ). The benefit to this retrieval method is that data validation is performed sooner, improving the probability of data correction being possible.

Weakly typed data retrieval allows for quick code writing, and allows for the data to be used in some fashion when the developer doesn't know beforehand what types will be returned. Further, with some effort, the programmer can extract the value into a variable of the proper type by using the GetFieldType or GetDataTypeName methods of the DataReader. **Common Errors**

A DataReader can in some cases be used in place of a DataTable, however many programmers have experienced connection bloat when following this approach. A DataReader can only be used against an (already) open database connection; that connection isn't closed until the DataReader's Dispose method is called. If an exception is thrown while the data is being processed, for example as described in Strong vs Weak Typing, above, the Dispose method will never be called if the developer writes code explicitly declaring and disposing the DataReader without the use of a try-finally block. The C# using construct is a good way to avoid this problem, as shown below in the code example.

**SQL DATA ADAPTER:**

In ADO.NET, a **DataAdapter** functions as a bridge between a data source, and a disconnected data class, such as a DataSet. At the simplest level it will specify SQL commands that provide elementary CRUD functionality. At a more advanced level it offers all the functions required in order to create Strongly Typed DataSets, including DataRelations. Data adapters are an integral part of ADO.NET managed providers, which are the set of objects used to communicate between a data source and a dataset. (In addition to adapters, managed providers include connection objects, data reader objects, and command objects.) Adapters are used to exchange data between a data source and a dataset. In many applications, this means reading data from a database into a dataset, and then writing changed data from the dataset back to the database. However, a data adapter can move data between any source and a dataset. For example, there could be an adapter that moves data between a Microsoft Exchange server and a dataset.

**SqlDataAdapter** interacts with the DataTable type. It can fill a DataTable with a table from your SQL Server database. Here we see a quick example. We then review important members (methods, events and properties) on SqlDataAdapter.

**DATA SET:**

Most commonly a dataset corresponds to the contents of a single database table, or a single statistical data matrix, where each column of the table represents a particular variable, and each row corresponds to a given member of the dataset in question. The dataset lists values for each of the variables, such as height and weight of an object, for each member of the dataset. Each value is known as a datum. The dataset may comprise data for one or more members, corresponding to the number of rows.

The term dataset may also be used more loosely, to refer to the data in a collection of closely related tables, corresponding to a particular experiment or event.

**6.0 SYSTEM TESTING**

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding. In fact, testing is the one step in the software engineering process that could be viewed as destructive rather than constructive.

A strategy for software testing integrates software test case design methods into a well-planned series of steps that result in the successful construction of software. Testing is the set of activities that can be planned in advance and conducted systematically. The underlying motivation of program testing is to affirm software quality with methods that can economically and effectively apply to both strategic to both large and small-scale systems.

The software engineering process can be viewed as a spiral. Initially system engineering defines the role of software and leads to software requirement analysis where the information domain, functions, behavior, performance, constraints and validation criteria for software are established. Moving inward along the spiral, we come to design and finally to coding. To develop computer software we spiral in along streamlines that decrease the level of abstraction on each turn. A strategy for software testing may also be viewed in the context of the spiral. Unit testing begins at the vertex of the spiral and concentrates on each unit of the software as implemented in source code. Testing progress by moving outward along the spiral to integration testing, where the focus is on the design and the construction of the software architecture. Talking another turn on outward on the spiral we encounter validation testing where requirements established as part of software requirements analysis are validated against the software that has been constructed. Finally we arrive at system testing, where the software and other system elements are tested as a whole.

* White-box testing,
* Black-box testing,
* Unit testing,
* Integration testing,
* System testing,
* Acceptance testing,
* Validation testing

**WHITE-BOX TESTING**

 White-box testing can be applied at the [unit](http://en.wikipedia.org/wiki/Unit_testing), [integration](http://en.wikipedia.org/wiki/Integration_testing) and [system](http://en.wikipedia.org/wiki/System_testing) levels of the **software testing** process, it is usually done at the unit level. It can test paths within a unit, paths between units during integration, and between subsystems during a system–level test. Though this method of test design can uncover many errors or problems, it might not detect unimplemented parts of the specification or missing requirements.

Techniques used in white-box testing include:

* [API](http://en.wikipedia.org/wiki/Application_programming_interface) testing (application programming interface) – testing of the application using public and private APIs
* [Code coverage](http://en.wikipedia.org/wiki/Code_coverage) – creating tests to satisfy some criteria of code coverage (e.g., the test designer can create tests to cause all statements in the program to be executed at least once)
* [Fault injection](http://en.wikipedia.org/wiki/Fault_injection) methods – intentionally introducing faults to gauge the efficacy of testing strategies
* [Mutation testing](http://en.wikipedia.org/wiki/Mutation_testing) methods
* [Static testing](http://en.wikipedia.org/wiki/Static_testing) methods

Code coverage tools can evaluate the completeness of a test suite that was created with any method, including black-box testing. This allows the software team to examine parts of a system that are rarely tested and ensures that the most important [function points](http://en.wikipedia.org/wiki/Function_points) have been tested. Code coverage as a [software metric](http://en.wikipedia.org/wiki/Software_metric) can be reported as a percentage for:

* *Function coverage*, which reports on functions executed
* *Statement coverage*, which reports on the number of lines executed to complete the test

100% statement coverage ensures that all code paths, or branches (in terms of [control flow](http://en.wikipedia.org/wiki/Control_flow)) are executed at least once. This is helpful in ensuring correct functionality, but not sufficient since the same code may process different inputs correctly or incorrectly.

**BLACK-BOX TESTING**

**Black-box testing** treats the software as a "black box", examining functionality without any knowledge of internal implementation. The tester is only aware of what the software is supposed to do, not how it does it. Black-box testing methods include: equivalence partitioning, boundary value analysis, all-pairs testing, state transition tables, decision table testing, fuzz testing, model-based testing, use case testing, exploratory testing and specification-based testing.

**Specification-based testing** aims to test the functionality of software according to the applicable requirements. This level of testing usually requires thorough test cases to be provided to the tester, who then can simply verify that for a given input, the output value (or behavior), either "is" or "is not" the same as the expected value specified in the test case. Test cases are built around specifications and requirements, i.e., what the application is supposed to do. It uses external descriptions of the software, including specifications, requirements, and designs to derive test cases. These tests can be functional or non-functional, though usually functional.

Specification-based testing may be necessary to assure correct functionality, but it is insufficient to guard against complex or high-risk situations.

One advantage of the black box technique is that no programming knowledge is required. Whatever biases the programmers may have had, the tester likely has a different set and may emphasize different areas of functionality. On the other hand, black-box testing has been said to be "like a walk in a dark labyrinth without a flashlight. Because they do not examine the source code, there are situations when a tester writes many test cases to check something that could have been tested by only one test case, or leaves some parts of the program untested. This method of test can be applied to all levels of software testing: unit, integration, system and acceptance. It typically comprises most if not all testing at higher levels, but can also dominate unit testing as well.

**UNIT TESTING**

Unit testing, also known as component testing, refers to tests that verify the functionality of a specific section of code, usually at the function level. In an object-oriented environment, this is usually at the class level, and the minimal unit tests include the constructors and destructors.

These types of tests are usually written by developers as they work on code (white-box style), to ensure that the specific function is working as expected. One function might have multiple tests, to catch corner cases or other branches in the code. Unit testing alone cannot verify the functionality of a piece of software, but rather is used to assure that the building blocks the software uses work independently of each other.

Unit testing is a software development process that involves synchronized application of a broad spectrum of defect prevention and detection strategies in order to reduce software development risks, time, and costs. It is performed by the software developer or engineer during the construction phase of the software development lifecycle. Rather than replace traditional QA focuses, it augments it. Unit testing aims to eliminate construction errors before code is promoted to QA; this strategy is intended to increase the quality of the resulting software as well as the efficiency of the overall development and QA process.

Depending on the organization's expectations for software development, unit testing might include static code analysis, data flow analysis metrics analysis, peer code reviews, code coverage analysis and other software verification practices.

**INTEGRATION TESTING:**

Integration testing is any type of software testing that seeks to verify the interfaces between components against a software design. Software components may be integrated in an iterative way or all together ("big bang"). Normally the former is considered a better practice since it allows interface issues to be located more quickly and fixed.

Integration testing works to expose defects in the interfaces and interaction between integrated components (modules). Progressively larger groups of tested software components corresponding to elements of the architectural design are integrated and tested until the software works as a system.

**SYSTEM TESTING:**

System testing tests a completely integrated system to verify that it meets its requirements.

In addition, the software testing should ensure that the program, as well as working as expected, does not also destroy or partially corrupt its operating environment or cause other processes within that environment to become inoperative (this includes not corrupting shared memory, not consuming or locking up excessive resources and leaving any parallel processes unharmed by its presence).

**ACCEPTANCE TESTING:**

The acceptance test suite is run against the supplied input data or using an acceptance test script to direct the testers. Then the results obtained are compared with the expected results. If there is a correct match for every case, the test suite is said to pass. If not, the system may either be rejected or accepted on conditions previously agreed between the sponsor and the manufacturer.

The objective is to provide confidence that the delivered system meets the business requirements of both sponsors and users. The acceptance phase may also act as the final quality gateway, where any quality defects not previously detected may be uncovered.

A principal purpose of acceptance testing is that, once completed successfully, and provided certain additional (contractually agreed) acceptance criteria are met, the sponsors will then sign off on the system as satisfying the contract (previously agreed between sponsor and manufacturer), and deliver final payment.

Acceptance testing can mean one of two things:

1. A [smoke test](http://en.wikipedia.org/wiki/Smoke_testing#Software_development) is used as an acceptance test prior to introducing a new build to the main testing process, i.e. before [integration](http://en.wikipedia.org/wiki/Integration_testing) or [regression](http://en.wikipedia.org/wiki/Regression_testing).
2. Acceptance testing performed by the customer, often in their lab environment on their own hardware, is known as [user acceptance testing](http://en.wikipedia.org/wiki/User_acceptance_testing) (UAT). Acceptance testing may be performed as part of the hand-off process between any two phases of development.

**VALIDATION TESTING:**

Validation is intended to check that development and verification procedures for a product, service, or system (or portion thereof, or set thereof) result in a product, service, or system (or portion thereof, or set thereof) that meets initial requirements. For a new development flow or verification flow, validation procedures may involve modeling either flow and using simulations to predict faults or gaps that might lead to invalid or incomplete verification or development of a product, service, or system (or portion thereof, or set thereof). A set of validation requirements, specifications, and regulations may then be used as a basis for qualifying a development flow or verification flow for a product, service, or system (or portion thereof, or set thereof). Additional validation procedures also include those that are designed specifically to ensure that modifications made to an existing qualified development flow or verification flow will have the effect of producing a product, service, or system (or portion thereof, or set thereof) that meets the initial design requirements, specifications, and regulations; these validations help to keep the flow qualified. It is a process of establishing evidence that provides a high degree of assurance that a product, service, or system accomplishes its intended requirements. This often involves acceptance of fitness for purpose with end users and other product stakeholders. This is often an external process.

**7. SYSTEM IMPLEMENTATION**

Implementation is the stage in the project where the theoretical design is turned into a working system. The most crucial stage is achieving a successful new system & giving the user confidence in that the new system will work efficiently & effectively in the implementation state.

**The stage consists of**

Testing the developed program with simple data.

Detection’s and correction of error.

Creating whether the system meets user requirements.

Testing whether the system.

Making necessary changes as desired by the user.

Training user personnel.

**7.1 Implementation Procedures**

The implementation phase is less creative than system design. A system project may be dropped at any time prior to implementation, although it becomes more difficult when it goes to the design phase.

The final report to the implementation phase includes procedural flowcharts, record layouts, report layouts, and a workable plan for implementing the candidate system design into an operational one. Conversion is one aspect of implementation.

**Several procedures of documents are unique to the conversion phase. They include the following,**

* The conversion portion of the implementation plan is finalized and approved.
* Files are converted.
* Parallel processing between the existing and the new system are logged on a special form.
* Assuming no problems, parallel processing is discontinued. Implementation results are documented for reference.

**7.2 USER TRAINING**

User Training is designed to prepare the user for testing & convening the system. There are several ways to train the user. They are

1) User Manual.

2) Help Screens.

3) Training Demonstration.

1) User Manual:

The summary of important functions about the system and software can be provided as a document to the user.

2) Help Screens:

This features now available in every software package, especially when it is used with a menu. The user selects the “Help” option from the menu. The system accesses the necessary description or information for user reference.

3) Training Demonstration:

Another User Training element is a Training Demonstration. Live demonstrations with personal contact are extremely effective for Training Users.

**7.3 OPERATIONAL DOCUMENTATION**

Documentation means of communication; it establishes design and performance criteria for phases of the project. Documentation is descriptive information that portrays the use and / or operation of the system.

(i) Documentation Tools

Document production and desktop publishing tool support nearly ever aspect of software developers. Most software development organizations spend a substantial amount of time developing documents, and in many cases the documentation process itself is quite inefficient. It is not use unusual for a software development effort on documentation. For this reason, documentation tools provide an important opportunity to improve productivity.

(ii) Document Restructuring

Creating document is far too timed consuming. If the system work’s, we’ll live with what we have. In some cases, this is the correct approach. It is not possible to recreate document for hundreds of computer programs.

Documentation must be updated, but we have limited resources. It may not be necessary to fully redocument an application. Rather, those portions of the system that are currently undergoing change are fully documented.

The system is business critical and must be fully redocumented. Even in this case, an intelligent approach is to pare documentation to an essential minimum.

**8. DESIGN AND DEVELOPMENT PROCESS**

##### Fundamental Design Concepts

System design is a “How to” approach to the creation of a new system. System design goes through 2 phases. They are

1. Logical Design

2. Physical design

Logical design reviews the present physical system, prepares input and output specification, makes edit security and control specification.

Physical design maps out the details of the physical system, plans the system implementation devise a test and implementation plan.

**Quality Assurance**

Quality assurance consists of the auditing and reporting functions of management. The goal of quality assurance is to provide management with the data necessary to be informed about product quality, thereby gaining insight and confidence that product quality is meeting its goal.

**Quality Assurance Goals**

**Correctness**

The extent to which the program meets system specifications and user objectives.

**Reliability**

The degree to which the system performs its intended functions overtime.

**Efficiency**

The amount of computer resources required by a program to perform a function.

**Usability**

The effort required learning and operating a system.

**Maintainability**

To use with which program errors are located and corrected.

**Testability**

The effort required testing a program to ensure its correct performance.

**Portability**

The ease of transporting a program from one hardware configuration to another.

**Accuracy**

The required position in input editing, computations and output.

**Generic Risks**

The general risk involves two categories:

☼ Uncertainty: The risk may or may not happen; that is, there are no 100% probable risks

☼ Loss:If the risk becomes a reality, unwanted consequences or losses will occur.

Different categories of risks are considered.

**Project risks:-**

Project risks identify potential budgetary, schedule, personnel like staffing and organization, resource, customer, and requirements problems and their impact on a software project.

**Technical risks:-**

Technical risks identify potential design, implementation, interface, verification, and maintenance problems.

**Security Technologies and Policies**

Any system developed should be secured and protected against possible hazards. Security measures are provided to prevent unauthorized access of the database at various levels. At uninterrupted power supply should be so that the power failure or voltage fluctuations will not erase the data in the files.

Password protection and simple procedures to change the unauthorized access are provided to the users. The system allows the user to enter the system for product management and order status entry only through login utility. The user will have to enter the user name and password.

A multi-layered security architecture comprising firewalls, filtering routers, encryption and digital certification must be ensured in this project in real time that order and payment details protected from unauthorized access. The customer can access this order status only by using his customer code and order number.

##### **9. SYSTEM MAINTENANCE**

Maintenance is actually the implementation of the review plan. As important as it is, many programmers and analysts are to perform or identify themselves with the maintenance effort. There are psychological, personality and professional reasons for this. Analysts and programmers spend far more time maintaining programs than they do writing them. Maintenance accounts for 50-80 percent of total system development.

**Maintenance is expensive. One way to reduce the maintenance costs are through maintenance management and software modification audits.**

* Maintenance is not as rewarding as exciting as developing systems. It is perceived as requiring neither skill not experience.
* Users are not fully cognizant of the maintenance problem or its high cost.
* Few tools and techniques are available for maintenance.
* A good test plan is lacking.
* Standards, procedures, and guidelines are poorly defined and enforced.
* Programs are often maintained without care for structure and documentation.
* There are minimal standards for maintenance.
* Programmers expect that they will not be in their current commitment by time their programs go into the maintenance cycle.

**10. CONCLUSION**

It is concluded that the application works well and executed in well manner. The application is tested very well and errors are properly debugged. This project can be used simultaneously and can access from more than 100000 users at a same time. Simultaneous login from more than one place is tested.

This project works according to the restrictions and rules and regulation. Further enhancements can be made to the application, so that this application functions very attractive and useful manner than the present one. The speed of the transactions become more enough now because of client server operation.

**11. FUTURE ENHANCEMENT**

All application has its own merits and demerits. Even this application has been developed in prompt manner.

Here some future enhancements are available.

* Can implement as a mobile application
* GPS Tracking system can be enabled
* Women Safety Can be ensured
* Can cancel the trip in the last minute
* Payment gateway can be integrated for money transaction
* Wallet payment can be implemented for virtual money transfer.

**12. APPENDIX.**

**12.1 Sample Screens.**

**Screen shots should taken by candidates only**

**Reason:**

**You can enter your name and project related information while taking Screen shots**

**12.2 Sample Code.**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.OleDb;

using System.Collections;

using System.Web.Configuration;

public partial class searchtravel : System.Web.UI.Page

{

OleDbConnection con;

OleDbCommand cmd;

string query;

ArrayList arr = new ArrayList();

string usname, ttimefrom, ttimeto;

int count;

protected void Page\_Load(object sender, EventArgs e)

{

lbluser.Text = Session["uname"].ToString();

}

public void data()

{

string connstring = WebConfigurationManager.ConnectionStrings["connect"].ConnectionString;

con = new OleDbConnection(connstring);

con.Open();

}

protected void Button1\_Click(object sender, EventArgs e)

{

data();

query = "select tdate from traveldet where tfrom='" + dropfrom.SelectedItem + "' and tto='" + dropto.SelectedItem + "'";

cmd = new OleDbCommand(query, con);

OleDbDataReader rd = cmd.ExecuteReader();

while (rd.Read())

{

arr.Add(rd[0].ToString());

}

rd.Close();

con.Close();

count = arr.Count;

if (count > 0)

{

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

{

int h = Convert.ToInt32(((Convert.ToDateTime(arr[i].ToString())) - (Convert.ToDateTime(System.DateTime.Today.ToShortDateString()))).TotalDays);

data();

query = "update traveldet set tcnt=" + h + " where tdate='" + arr[i].ToString() + "' and tfrom='" + dropfrom.SelectedItem + "' and tto='" + dropto.SelectedItem + "'";

cmd = new OleDbCommand(query, con);

cmd.ExecuteNonQuery();

con.Close();

}

}

GridView1.Visible = true;

GridView1.DataBind();

lblack.Visible = false;

Panel1.Visible = false;

}

protected void GridView1\_SelectedIndexChanged(object sender, EventArgs e)

{

Panel1.Visible = true;

usname = GridView1.SelectedRow.Cells[0].Text.ToString();

usern.Text = GridView1.SelectedRow.Cells[0].Text.ToString();

// lblname.Text = GridView1.SelectedRow.Cells[1].Text.ToString();

lblgender.Text = GridView1.SelectedRow.Cells[1].Text.ToString();

lblage.Text = GridView1.SelectedRow.Cells[2].Text.ToString();

ttimefrom = GridView1.SelectedRow.Cells[4].Text.ToString();

ttimeto = GridView1.SelectedRow.Cells[5].Text.ToString();

data();

query = "select toccupation,toccutype,tname from signupdet where uname='" + usname + "'";

cmd = new OleDbCommand(query, con);

OleDbDataReader rd = cmd.ExecuteReader();

while (rd.Read())

{

lbloccupation.Text = rd[0].ToString();

lbloccutype.Text = rd[1].ToString();

lblname.Text = rd[2].ToString();

}

rd.Close();

con.Close();

data();

query = "select \* from traveldet where uname='" + usname + "' and tfromtime='" + ttimefrom + "' and ttotime='" + ttimeto + "'";

cmd = new OleDbCommand(query, con);

OleDbDataReader rd1 = cmd.ExecuteReader();

while (rd1.Read())

{

lblvehiclemake.Text = rd1[1].ToString();

lblvehiclemodel.Text = rd1[2].ToString();

lblfueltype.Text = rd1[3].ToString();

lblseater.Text = rd1[4].ToString();

lblfrom.Text = rd1[5].ToString();

lblvia.Text = rd1[6].ToString();

lblto.Text = rd1[7].ToString();

lbltraveldate.Text = rd1[8].ToString();

lblfromtime.Text = rd1[9].ToString();

lbltotime.Text = rd1[11].ToString();

lbllicnum.Text = rd1[20].ToString();

}

rd1.Close();

con.Close();

}

protected void Button2\_Click(object sender, EventArgs e)

{

data();

query = "select tname,tage,tgender,toccupation,toccutype,tphone,temail,st1,st2,city,state,pinc from signupdet where uname='" + lbluser.Text + "'";

cmd = new OleDbCommand(query, con);

OleDbDataReader rd = cmd.ExecuteReader();

while (rd.Read())

{

string address = rd[7].ToString() + "," + rd[8].ToString() + "," + rd[9].ToString() + "," + rd[10].ToString() + "," + rd[11].ToString();

data();

query = "insert into requestdet(fromuname,touname,vmake,vmodel,ftype,seater,trafrom,travia,trato,tradate,tratimefrom,tratimeto,status,tname,tage,tgender,toccu,typeoccu,tphone,temail,taddress,tlicnum)values('" + lbluser.Text + "','" + usern.Text + "','" + lblvehiclemake.Text + "','" + lblvehiclemodel.Text + "','" + lblfueltype.Text + "','" + lblseater.Text + "','" + lblfrom.Text + "','" + lblvia.Text + "','" + lblto.Text + "','" + lbltraveldate.Text + "','" + lblfromtime.Text + "','" + lbltotime.Text + "','NO','" + rd[0].ToString() + "','" + rd[1].ToString() + "','" + rd[2].ToString() + "','" + rd[3].ToString() + "','" + rd[4].ToString() + "','" + rd[5].ToString() + "','" + rd[6].ToString() + "','" + address + "','" + lbllicnum.Text + "')";

cmd = new OleDbCommand(query, con);

cmd.ExecuteNonQuery();

con.Close();

}

rd.Close();

con.Close();

lblack.Visible = true;

Panel1.Visible = false;

}

}

using System;

using System.Collections.Generic;

using System.Linq;

using System.Web;

using System.Web.UI;

using System.Web.UI.WebControls;

using System.Data.OleDb;

using System.Web.Configuration;

public partial class usersignup : System.Web.UI.Page

{

OleDbConnection con;

OleDbCommand cmd;

string query,gender;

protected void Page\_Load(object sender, EventArgs e)

{

}

public void data()

{

string connstring = WebConfigurationManager.ConnectionStrings["connect"].ConnectionString;

con = new OleDbConnection(connstring);

con.Open();

}

protected void Button1\_Click(object sender, EventArgs e)

{

if (RadioButton1.Checked == true)

{

gender = "Male";

}

else

{

gender = "Female";

}

data();

query = "insert into signupdet(tname,tgender,tage,tphone,temail,toccupation,toccutype,st1,st2,city,state,pinc,uname,pwd)values('" + txtname.Text + "','" + gender + "','" + txtage.Text + "','" + txtphno.Text + "','" + txtemail.Text + "','" + txtoccupation.Text + "','" + txttypeofoccu.Text + "','" + txtstreet1.Text + "','" + txtstreet2.Text + "','" + txtcity.Text + "','" + txtstate.Text + "','" + txtpin.Text + "','" + txtuname.Text + "','" + txtpwd.Text + "')";

cmd = new OleDbCommand(query, con);

cmd.ExecuteNonQuery();

con.Close();

lblack.Visible = true;

txtage.Text = "";

txtuname.Text = "";

txtcity.Text = "";

txtemail.Text = "";

txtname.Text = "";

txtoccupation.Text = "";

txtphno.Text = "";

txtpin.Text = "";

txtstate.Text = "";

txtstreet1.Text = "";

txtstreet2.Text = "";

txttypeofoccu.Text = "";

}

protected void txtuname\_TextChanged(object sender, EventArgs e)

{

data();

query = "select uname from signupdet where uname='" + txtuname.Text + "'";

cmd = new OleDbCommand(query, con);

OleDbDataReader rd = cmd.ExecuteReader();

if (rd.Read())

{

lblerr.Visible = true;

lblerr.Text = "Username Exists";

Button1.Enabled = false;

}

else

{

lblerr.Visible = false;

Button1.Enabled = true;

}

rd.Close();

con.Close();

}

}