Factory Statistics

ABSTRACT

Factory Statistics System is a web tool for maintaining factories in the globe. This system has been designed to maintain a comprehensive database of factories, including the details of workers employed, accidents, poisoning and diseases, safety measures adopted, etc. Monitor implementation of Factories Act .Produce statistical reports on the functioning of factories.

These Applications provide information you won't find elsewhere because these companies are not obligated to report this information to anyone. We provide profiles on more private companies than any other information provider on the web.

Public Company Profiles - Every public company worldwide is profiled in the Goliath database. We are unique in that we offer profiles by company and by subsidiaries within those companies, vs. our competitors who provide profiles by individual office. Our profiles aggregate offices by subsidiary to save you money and time.

INDEX

S. No CONTENTS

9. BIBILIOGRAPHY

Introduction

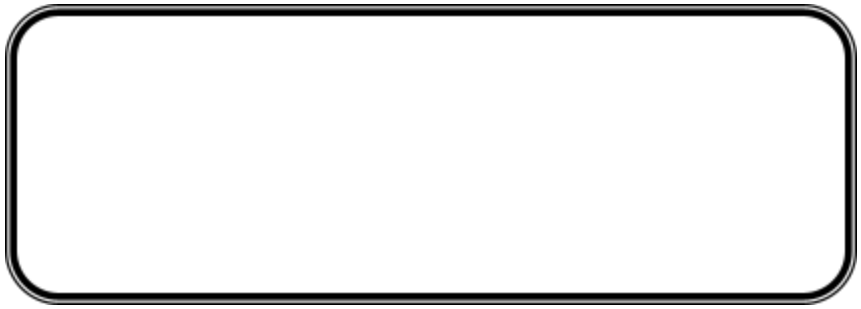
INTRODUCTION:

Factory Statistics System consists of list of records about the management of the factories .This is a web-based application. The project has three modules namely-Administrator, Reports, Factory information.

As the modern organizations are automated and computers are working as per the instructions, it becomes essential for the coordination of human beings, commodity and computers in a modern organization. Administrators of the project need to maintain the information of the factories very accurately and up to date. This information helps the other officer to handle and complete their works very efficiently.

The administrators and all the others can communicate with the system through this projects, thus facilitating effective implementation and monitoring of various activities of the project. This project consists the details of various factories.

In this system we are maintaining the Director login and how to factory registration, no of factories, Add accident & Inception, closed &Remove factories, Add conventions, and Add annual returns.



SYSTEM ANALYSIS:

Existing System:

The present system deals with day to day work of factory system. So monitoring the stock levels is very difficulty. Maintain the necessary activity schedule of different maintenance works in the factories and other related structures is also very difficult. It is very difficulties calculate the amount of work to be done under each maintenance cost of various activities.Retrieving previous data is quite difficult and also the cost involved in maintaining the records is very costly. So computerization is very much necessary.

Proposed System:

The proposed system maintains the necessary activity schedule of different maintenance works in the factories and other related structures. It also keeps track of various maintenance activities carried out in the past by the responsible maintenance authority at various levels. This System will also facilitate the user to draw a schematic and calculate the amount of work to be done under each maintenance cost of various activities and facilitates the user to add, modify, query, delete and prioritize the maintenance activities

Objective of The System:

This system has been designed to maintain a comprehensive database of factories, including the details of workers employed, accidents, poisoning and diseases, safety measures adopted,

etc. Monitor implementation of Factories Act .Produce statistical reports on the functioning of factories.

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 XP
- **Programming language: C#**
- **Web-Technology:** ASP.NET 2.0
- **Back-End:** SQL-SERVER 2005

- **Web Server:** IIS

.

Design

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.

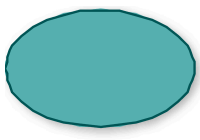
UML Diagrams:

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:

- Use case Diagram
- Sequence Diagram
- Collaboration Diagram
- Activity Diagram
- State chat Diagram

USECASE 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.

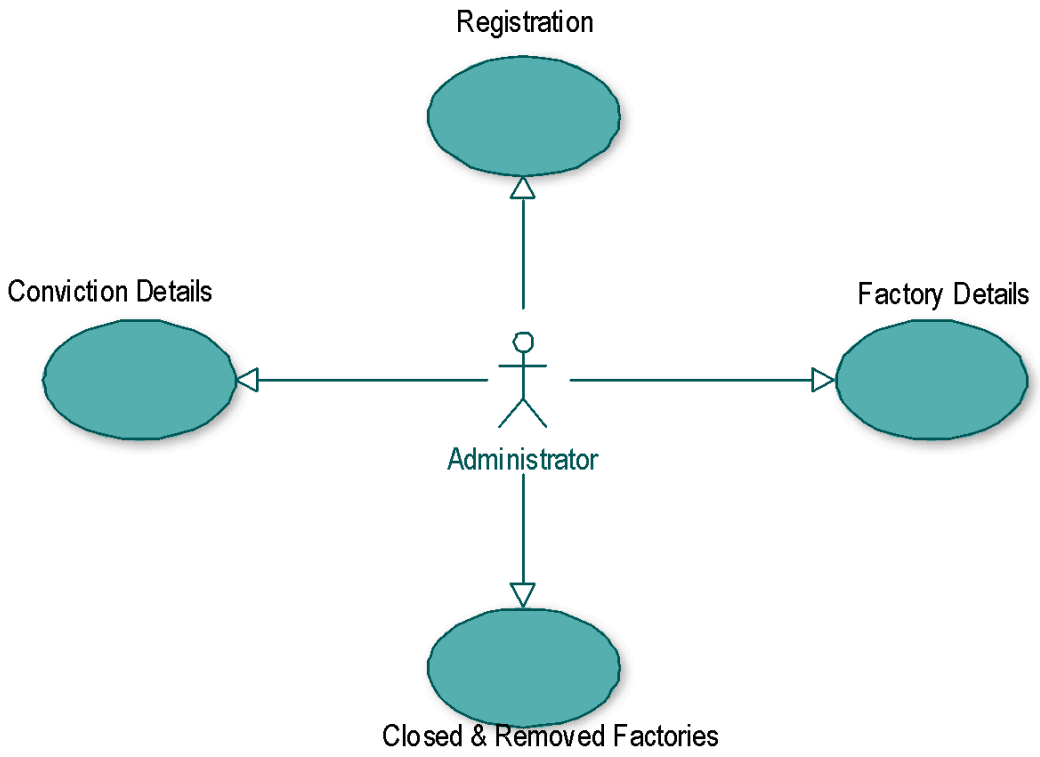
Use case diagram can be useful for getting an overall view of the system and clarifying who can do and more importantly what they can't do.

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.

USECASE DIAGRAM:

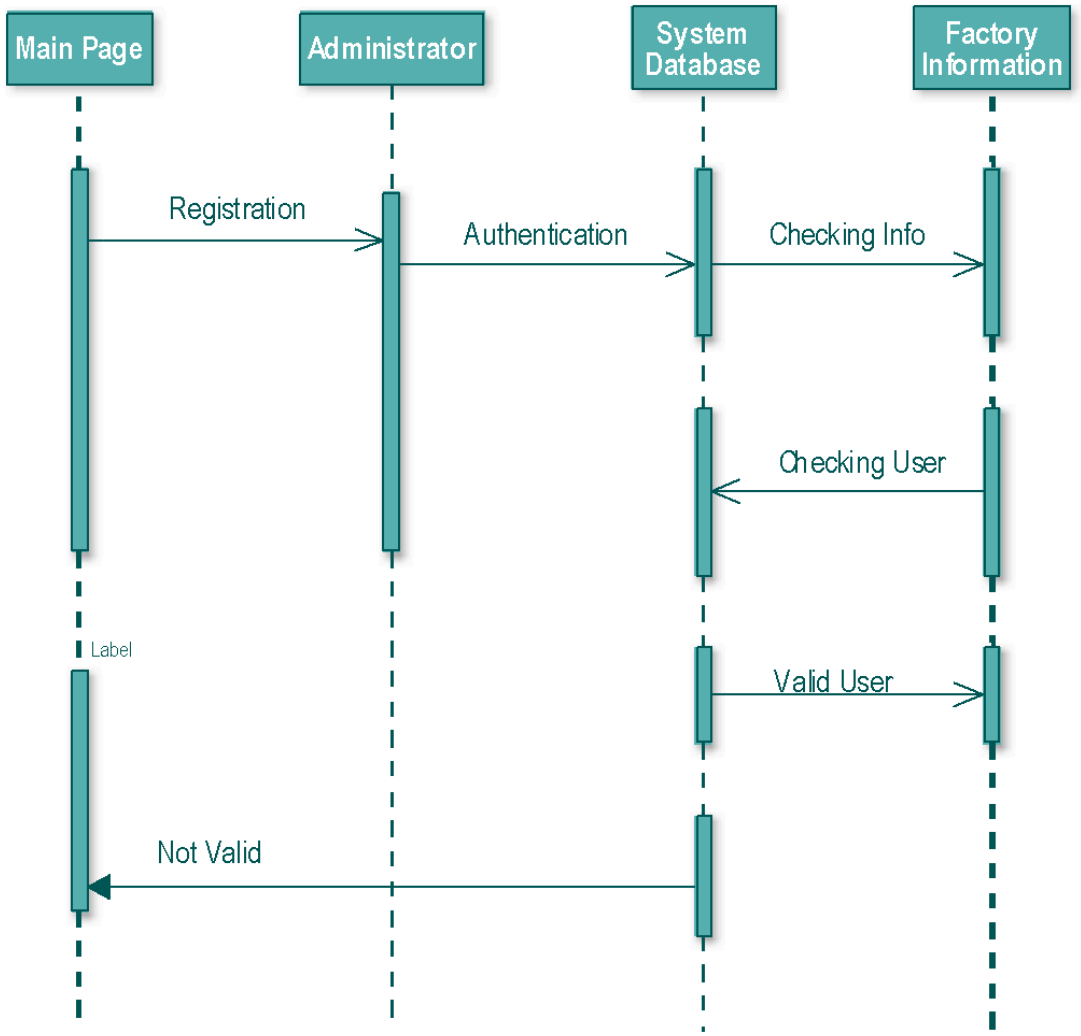
A Use case is a description of set of sequence of actions. Graphically it is rendered as an ellipse with solid line including only its name. Use case diagram is a behavioral diagram that shows a set of use cases and actors and their relationship. It is an association between the use cases and actors.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



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:

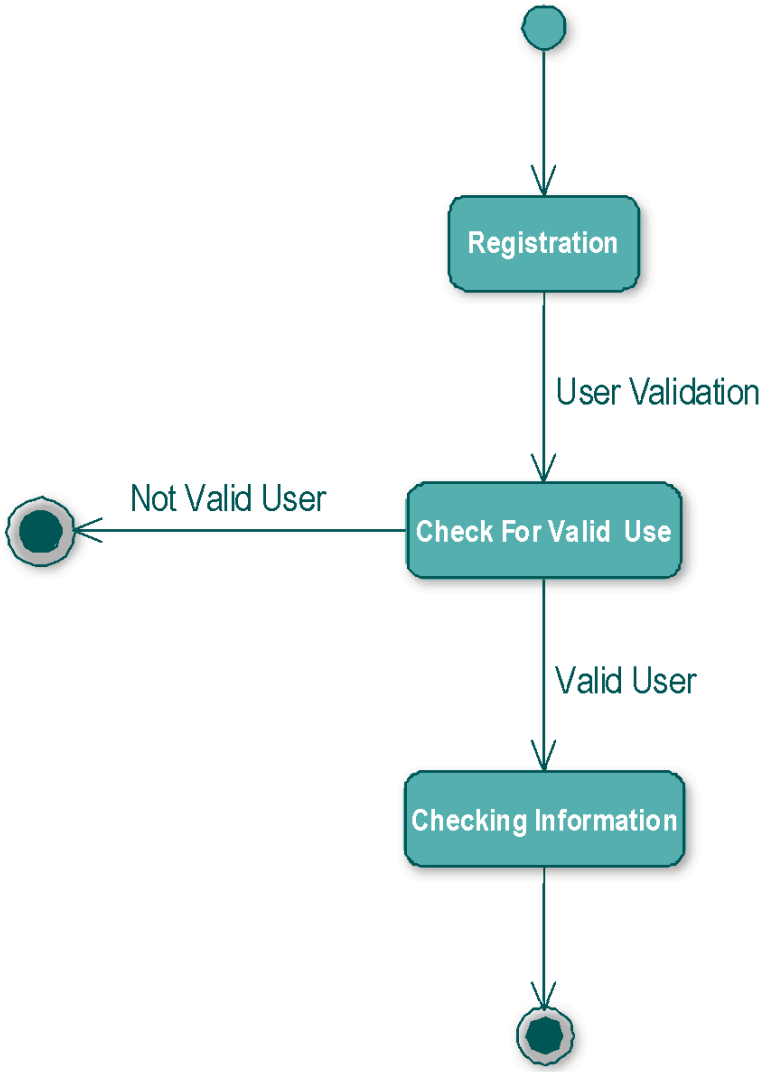
Class is nothing but a structure that contains both variables and methods. The Class Diagram shows a set of classes, interfaces, and collaborations and their relating ships. There is most common diagram in modeling the object oriented systems and are used to give the static view of a system. It shows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

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:

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

PROJECT MODULES

There are seven modules involved in this project:

- **Administration**
- **Reports**
- **Factory Info**

MODULE DESCRIPTION

Name of the module-1: Administration

Description: This module deals with registering the director or the administrator where he can process and update factory info regularly.

Submodules:

- Registration
- **Factories List**
- **Add Or Remove Factories**

Registration:

In this sub module the admin/director must be register. Only the register users can access the information for others the access will be denied.

Factories List:

In this sub module the list of factories is displayed. The admin will update the factories list.

Add or Remove Factories:

In this sub module the admin will remove the closed factories and will add the newely opened factories.

Name of the module-2: Reports

Description:

This module helps in having detailed statistical reports of the factory based on the performance level or time period and profit based charts.

Reports:

- List of Factories registered and workers employed in these factories.
- List of factories carrying on dangerous processes and the no. of workers employed in these factories.
- Statement showing the no. of working factories and the employment in these factories.
- Statement of distribution of working factories submitting the returns and their working strength.
- Statement of the distribution of working factories submitting the returns based on the no. of days worked.
- Statement of the fatal and non fatal accidents in factories.
- Statement of accidents by causes.
- Statement of accidents by age and sex
- Statement of Number of cases of poisoning & disease notified under section 89
- Statement of distribution of factories and employment based on the average no. of hours/week worked.
- Statement of inspections carried out.
- Statement of convictions.
- Statement showing the facilities available at each factory like the rest room, ambulance, canteens etc.

Name of the module-3: Factory Info

Description:

This module is the main module where we maintain the total information of different factories like profit/ loss since 5 years ,shares info,etc,.

Submodules: .

- Details of annual returns.

- Details of half yearly returns
- Convictions
- Inspections carried out at different factories.

Details Of Annual Returns:

In this submodule the details of annual returns of a factory are displayed. Based on this we can know the condition of the factory whether it is in profit/loss.

Details Of Half Yearly Returns:

In this submodule the details of annual returns of a factory are displayed. Based on this we can know the condition of the factory whether it is in profit/loss.

Convictions:

In this submodule we can see the conviction details of a certain factory.

OVERVIEW OF TECHNOLOGIES USED

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.

Features of the Common Language Runtime

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

With regards to security, managed components are awarded varying degrees of trust, depending on a number of factors that include their origin (such as the Internet, enterprise network, or local computer). This means that a managed component might or might not be able to perform file-access operations, registry-access operations, or other sensitive functions, even if it is being used in the same active application.

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

The runtime also enforces code robustness by implementing a strict type- and code-verification infrastructure called the common type system (CTS). The CTS ensures that all managed code is self-describing. The various Microsoft and third-party language compilers generate managed code that conforms to the CTS. This means that managed code can consume other managed types and instances, while strictly enforcing type fidelity and type safety.

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

The runtime also accelerates developer productivity. For example, programmers can write applications in their development language of choice, yet take full advantage of the runtime, the class library, and components written in other languages by other developers. Any compiler vendor who

chooses to target the runtime can do so. Language compilers that target the .NET Framework make the features of the .NET Framework available to existing code written in that language, greatly easing the migration process for existing applications.

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

The runtime is designed to enhance performance. Although the common language runtime provides many standard runtime services, managed code is never interpreted. A feature called just-in-time (JIT) compiling enables all managed code to run in the native machine language of the system on which it is executing. Meanwhile, the memory manager removes the possibilities of fragmented memory and increases memory locality-of-reference to further increase performance.

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

.NET Framework Class Library

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

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

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

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

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

Client Application Development

Client applications are the closest to a traditional style of application in Windows-based programming. These are the types of applications that display windows or forms on the desktop, enabling a user to perform a task. Client applications include applications such as

word processors and spreadsheets, as well as custom business applications such as data-entry tools, reporting tools, and so on. Client applications usually employ windows, menus,

buttons, and other GUI elements, and they likely access local resources such as the file system and peripherals such as printers.

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

In the past, developers created such applications using C/C++ in conjunction with the Microsoft Foundation Classes (MFC) or with a rapid application development (RAD) environment such as Microsoft® Visual Basic®. The .NET Framework incorporates aspects of these existing products into a single, consistent development environment that drastically simplifies the development of client applications. The Windows Forms classes contained in the .NET Framework are designed to be used for GUI development. You can easily create command windows, buttons, menus, toolbars, and other screen elements with the flexibility necessary to accommodate shifting business needs.

For example, the .NET Framework provides simple properties to adjust visual attributes associated with forms. In some cases the underlying operating system does not support changing these attributes directly, and in these cases the .NET Framework automatically recreates the forms. This is one of many ways in which the .NET Framework integrates the developer interface, making coding simpler and more consistent.

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

Server Application Development

Server-side applications in the managed world are implemented through runtime hosts. Unmanaged applications host the common language runtime, which allows your custom managed code to control the behavior of the server. This model provides you with all the features of the common language runtime and class library while gaining the performance and scalability of the host server.

The following illustration shows a basic network schema with managed code running in different server environments. Servers such as IIS and SQL Server can perform standard operations while your application logic executes through the managed code.

Server-side managed code

ASP.NET is the hosting environment that enables developers to use the .NET Framework to target Web-based applications. However, ASP.NET is more than just a runtime host; it is a complete architecture for developing Web sites and Internet-distributed objects using managed code. Both Web Forms and XML Web services use IIS and ASP.NET as the publishing mechanism for applications, and both have a collection of supporting classes in the .NET Framework.

XML Web services, an important evolution in Web-based technology, are distributed, server-side application components similar to common Web sites. However, unlike Web-based applications, XML Web services components have no UI and are not targeted for browsers such as Internet Explorer and Netscape Navigator. Instead, XML Web services consist of reusable software components designed to be consumed by other applications, such as traditional client applications, Web-based applications, or even other XML Web services. As a result, XML Web services technology is rapidly moving application development and deployment into the highly distributed environment of the Internet.

If you have used earlier versions of ASP technology, you will immediately notice the improvements that ASP.NET and Web Forms offers. For example, you can develop Web Forms pages

in any language that supports the .NET Framework. In addition, your code no longer needs to share the same file with your HTTP text (although it can continue to do so if you prefer). Web Forms pages execute in native machine language because, like any other managed application, they take full advantage of the runtime. In contrast, unmanaged ASP pages are always scripted and interpreted. ASP.NET pages are faster, more functional, and easier to develop than unmanaged ASP pages because they interact with the runtime like any managed application.

The .NET Framework also provides a collection of classes and tools to aid in development and consumption of XML Web services applications. XML Web services are built on standards such as SOAP (a remote procedure-call protocol), XML (an extensible data format), and WSDL (the Web Services Description Language). The .NET Framework is built on these standards to promote interoperability with non-Microsoft solutions.

For example, the Web Services Description Language tool included with the .NET Framework SDK can query an XML Web service published on the Web, parse its WSDL description, and produce C# or Visual Basic source code that your application can use to become a client of the XML Web service. The source code can create classes derived from classes in the class library that handle all the underlying communication using SOAP and XML parsing. Although you can use the class library to consume XML Web services directly, the Web Services Description Language tool and the other tools contained in the SDK facilitate your development efforts with the .NET Framework.

If you develop and publish your own XML Web service, the .NET Framework provides a set of classes that conform to all the underlying communication standards, such as SOAP, WSDL, and XML. Using those classes enables you to focus on the logic of your service, without concerning yourself with the communications infrastructure required by distributed software development.

Finally, like Web Forms pages in the managed environment, your XML Web service will run with the speed of native machine language using the scalable communication of IIS.

Active Server Pages.NET

ASP.NET is a programming framework built on the common language runtime that can be used on a server to build powerful Web applications. ASP.NET offers several important advantages over previous Web development models:

- **Enhanced Performance.** ASP.NET is compiled common language runtime code running on the server. Unlike its interpreted predecessors, ASP.NET can take advantage of early binding, just-in-time compilation, native optimization, and caching services right out of the box. This amounts to dramatically better performance before you ever write a line of code.

- **World-Class Tool Support.** The ASP.NET framework is complemented by a rich toolbox and designer in the Visual Studio integrated development environment. WYSIWYG editing, drag-and-drop server controls, and automatic deployment are just a few of the features this powerful tool provides.

- **Power and Flexibility.** Because ASP.NET is based on the common language runtime, the power and flexibility of that entire platform is available to Web application developers. The .NET Framework class library, Messaging, and Data Access solutions are all seamlessly accessible from the Web. ASP.NET is also language-independent, so you can choose the language that best applies to your application or partition your application across many languages. Further, common language runtime interoperability guarantees that your existing investment in COM-based development is preserved when migrating to ASP.NET.

- **Simplicity.** ASP.NET makes it easy to perform common tasks, from simple form submission and client authentication to deployment and site configuration. For example, the ASP.NET page framework allows you to build user interfaces that cleanly separate application logic from presentation code and to handle events in a simple, Visual Basic - like forms processing model. Additionally, the common language runtime simplifies development, with managed code services such as automatic reference counting and garbage collection.

- **Manageability.** ASP.NET employs a text-based, hierarchical configuration system, which simplifies applying settings to your server environment and Web applications. Because configuration information is stored as plain text, new settings may be applied without the aid of local administration tools. This "zero local administration" philosophy extends to deploying ASP.NET Framework applications as well. An ASP.NET Framework application is deployed to a server simply by copying the necessary files to the server. No server restart is required, even to deploy or replace running compiled code.

- **Scalability and Availability.** ASP.NET has been designed with scalability in mind, with features specifically tailored to improve performance in clustered and multiprocessor environments. Further, processes are closely monitored and managed

By the ASP.NET runtime, so that if one misbehaves (leaks, deadlocks), a new process can be created in its place, which helps keep your application constantly available to handle requests.

- **Customizability and Extensibility.** ASP.NET delivers a well-factored architecture that allows developers to "plug-in" their code at the appropriate level. In fact, it is possible to extend or replace any subcomponent of the ASP.NET runtime with your own custom-written component. Implementing custom authentication or state services has never been easier.

- **Security.** With built in Windows authentication and per-application configuration, you can be assured that your applications are secure.

Language Support

The Microsoft .NET Platform currently offers built-in support for three languages: C#, Visual Basic, and JScript.

What is ASP.NET Web Forms?

The ASP.NET Web Forms page framework is a scalable common language runtime programming model that can be used on the server to dynamically generate Web pages.

Intended as a logical evolution of ASP (ASP.NET provides syntax compatibility with existing pages), the ASP.NET Web Forms framework has been specifically designed to address a number of key deficiencies in the previous model. In particular, it provides:

- The ability to create and use reusable UI controls that can encapsulate common functionality and thus reduce the amount of code that a page developer has to write.
- The ability for developers to cleanly structure their page logic in an orderly fashion (not "spaghetti code").
- The ability for development tools to provide strong WYSIWYG design support for pages (existing ASP code is opaque to tools).

ASP.NET Web Forms pages are text files with an .aspx file name extension. They can be deployed throughout an IIS virtual root directory tree. When a browser client requests .aspx resources, the ASP.NET runtime parses and compiles the target file into a .NET Framework class. This class can then be used to dynamically process incoming requests. (Note that the .aspx file is compiled only the first time it is accessed; the compiled type instance is then reused across multiple requests).

An ASP.NET page can be created simply by taking an existing HTML file and changing its file name extension to .aspx (no modification of code is required). For example, the following sample demonstrates a simple HTML page that collects a user's name and category preference and then performs a form post back to the originating page when a button is clicked:

ASP.NET provides syntax compatibility with existing ASP pages. This includes support for <% %> code render blocks that can be intermixed with HTML content within an .aspx file. These code blocks execute in a top-down manner at page render time.

Code-Behind Web Forms

ASP.NET supports two methods of authoring dynamic pages. The first is the method shown in the preceding samples, where the page code is physically declared within the originating .aspx file. An alternative approach--known as the code-behind method--enables the page code to be more cleanly separated from the HTML content into an entirely separate file.

Introduction to ASP.NET Server Controls

In addition to (or instead of) using <% %> code blocks to program dynamic content, ASP.NET page developers can use ASP.NET server controls to program Web pages. Server controls are declared within an .aspx file using custom tags or intrinsic HTML tags that contain a **runat="server"** attributes value. Intrinsic HTML tags are handled by one of the controls in the **System.Web.UI.HtmlControls** namespace. Any tag that doesn't explicitly map to one of the controls is assigned the type of **System.Web.UI.HtmlControls.HtmlGenericControl**.

Server controls automatically maintain any client-entered values between round trips to the server. This control state is not stored on the server (it is instead stored within an **<input type="hidden">** form field that is round-tripped between requests). Note also that no client-side script is required.

In addition to supporting standard HTML input controls, ASP.NET enables developers to utilize richer custom controls on their pages. For example, the following sample demonstrates how the **<asp:adrotator>** control can be used to dynamically display rotating ads on a page.

1. ASP.NET Web Forms provide an easy and powerful way to build dynamic Web UI.
2. ASP.NET Web Forms pages can target any browser client (there are no script library or cookie requirements).
3. ASP.NET Web Forms pages provide syntax compatibility with existing ASP pages.
4. ASP.NET server controls provide an easy way to encapsulate common functionality.
5. ASP.NET ships with 45 built-in server controls. Developers can also use controls built by third parties.
6. ASP.NET server controls can automatically project both up level and down level HTML.
7. ASP.NET templates provide an easy way to customize the look and feel of list server controls.
8. ASP.NET validation controls provide an easy way to do declarative client or server data validation.

Crystal Reports

Crystal Reports for Visual Basic .NET is the standard reporting tool for Visual Basic.NET; it brings the ability to create interactive, presentation-quality content — which has been the strength of Crystal Reports for years — to the .NET platform.

With Crystal Reports for Visual Basic.NET, you can host reports on Web and Windows platforms and publish Crystal reports as Report Web Services on a Web server.

To present data to users, you could write code to loop through record sets and print them inside your Windows or Web application. However, any work beyond basic formatting can be complicated: consolidations, multiple level totals, charting, and conditional formatting are difficult to program.

With Crystal Reports for Visual Studio .NET, you can quickly create complex and professional-looking reports. Instead of coding, you use the Crystal Report Designer interface to create and format the report you need. The powerful Report Engine processes the formatting, grouping, and charting criteria you specify.

Report Experts

Using the Crystal Report Experts, you can quickly create reports based on your development needs:

- Choose from report layout options ranging from standard reports to form letters, or build your own report from scratch.
- Display charts that users can drill down on to view detailed report data.
- Calculate summaries, subtotals, and percentages on grouped data.
- Show TopN or BottomN results of data.
- Conditionally format text and rotate text objects.

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.

Relational Database

There are different ways to organize data in a database but relational databases are one of the most effective. Relational database systems are an application of mathematical set theory to the problem of effectively organizing data. In a relational database, data is collected into tables (called relations in relational theory).

When organizing data into tables, you can usually find many different ways to define tables. Relational database theory defines a process, normalization, which ensures that the set of tables you define will organize your data effect

Client/Server:-

In a client/server system, the server is a relatively large computer in a central location that manages a resource used by many people. When individuals need to use the resource, they connect over the network from their computers, or clients, to the server.

Examples of servers are: In a client/server database architecture, the database files and DBMS software reside on a server. A communications component is provided so applications can run on separate clients and communicate to the database server over a network. The SQL Server communication component also allows communication between an application running on the server and SQL Server.

Server applications are usually capable of working with several clients at the same time. SQL Server can work with thousands of client applications simultaneously. The server has features to prevent the logical problems that occur if a user tries t read or modify data currently being used by others.

While SQL Server is designed to work as a server in a client/server network, it is also capable of working as a stand-alone database directly on the client. The scalability and ease-of-use features of SQL Server allow it to work efficiently on a client without consuming too many resources.

Structured Query Language (SQL)

To work with data in a database, you must use a set of commands and statements (language) defined by the DBMS software. There are several different languages that can be used with relational databases; the most common is SQL. Both the American National Standards Institute (ANSI) and the International Standards Organization (ISO) have defined standards for SQL. Most modern DBMS products support the Entry Level of SQL-92, the latest SQL standard (published in 1992).

SQL Server Features

Microsoft SQL Server supports a set of features that result in the following benefits:

Ease of installation, deployment, and use

SQL Server includes a set of administrative and development tools that improve your ability to install, deploy, manage, and use SQL Server across several sites.

Scalability

The same database engine can be used across platforms ranging from laptop computers running Microsoft Windows® 95/98 to large, multiprocessor servers running Microsoft Windows NT®, Enterprise Edition.

Data warehousing

SQL Server includes tools for extracting and analyzing summary data for online analytical processing (OLAP). SQL Server also includes tools for visually designing databases and analyzing data using English-based questions.

System integration with other server software

SQL Server integrates with e-mail, the Internet, and Windows.

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, and each database can store either interrelated data or data unrelated to that in the 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 that are 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.

Normalization theory:

Relations are to be normalized to avoid anomalies. In insert, update and delete operations. Normalization theory is built around the concept of normal forms. A relation is said to be in a particular form if it satisfies a certain specified set if constraints. To decide a suitable logical structure for given database design the concept of normalization, which are briefly described below.

1. 1 st Normal Form (1 N.F): A relation is said to be in 1 NF is and only if all unaligned domains contain values only. That is the fields of an n-set should have no group items and no repeating groups.
2. 2 nd Normal Form (2 N.F) : A relation is said to be in 2 NF is and only if it is in 1 NF and every non key attribute is fully dependent on primary key. This normal takes care of functional dependencies on non-key attributes.

3. 3 rd Normal Form (3 N.F) : A relation is said to be in 3 NF is and only if it is in 2 NF and every non key attribute is non transitively dependent on the primary key. This normal form avoids the transitive dependencies on the primary key.

4. Boyce code Normal Form (BCNF) : This is a stronger definition than that of NF. A relation is said to be in BCNF if and only if every determinant is a Candidate key.

5. 4 th Normal Form (4 NF) : A relation is said to be in 4 NF if and only if whenever there exists a multi valued dependency in a relation say A->->B then all of the relation are also functionally dependent on A(i.e. A->X for all attributes x of the relation.).

6. 5 th Normal Form (5 NF) OR Projection Join Normal Form (PJNF): A relation R is in 5 NF .if and only if every join dependency in R is implied by the candidate key on R . A relation can't be non-loss split into two tables but can be split into three tables. This is called Join Dependency.

1.3 Middleware Technology

Activex Data Objects.Net Overview

ADO.NET is an evolution of the ADO data access model that directly addresses user requirements for developing scalable applications. It was designed specifically for the web with scalability, statelessness, and XML in mind.

ADO.NET uses some ADO objects, such as the Connection and Command objects, and also introduces new objects. Key new ADO.NET objects include the Dataset, Data Reader, and Data Adapter.

The important distinction between this evolved stage of ADO.NET and previous data architectures is that there exists an object -- the Dataset -- that is separate and distinct from any data stores. Because of that, the Dataset functions as a standalone entity. You can think of the Dataset as an always disconnected record set that knows nothing about the source or destination of the data it contains. Inside a Dataset, much like in a database, there are tables, columns, relationships, constraints, views, and so forth.

A Data Adapter is the object that connects to the database to fill the Dataset. Then, it connects back to the database to update the data there, based on operations performed while the Dataset held the data. In the past, data processing has been primarily connection-based. Now, in an effort to make multi-tiered apps more efficient, data processing is turning to a message-based approach that revolves around chunks of information. At the center of this approach is the Data Adapter, which provides a bridge to retrieve and save data between a Dataset and its source data store. It accomplishes this by means of requests to the appropriate SQL commands made against the data store.

The XML-based Dataset object provides a consistent programming model that works with all models of data storage: flat, relational, and hierarchical. It does this by having no 'knowledge'

of the source of its data, and by representing the data that it holds as collections and data types. No matter what the source of the data within the Dataset is, it is manipulated through the same set of standard APIs exposed through the Dataset and its subordinate objects.

While the Dataset has no knowledge of the source of its data, the managed provider has detailed and specific information. The role of the managed provider is to connect, fill, and persist the Dataset to and from data stores. The OLE DB and SQL Server .NET Data Providers (System.Data.OleDb and System.Data.SqlClient) that are part of the .Net Framework provide four basic objects: the Command, Connection, Data Reader and Data Adapter. In the remaining sections of this document, we'll walk through each part of the Dataset and the OLE DB/SQL Server .NET Data Providers explaining what they are, and how to program against them. The following sections will introduce you to some objects that have evolved, and some that are new. These objects are:

- Connections. For connection to and managing transactions against a database.
- Commands. For issuing SQL commands against a database.
- Data Readers. For reading a forward-only stream of data records from a SQL Server data source.
- Datasets. For storing, removing and programming against flat data, XML data and relational data.
- Data Adapters. For pushing data into a Dataset, and reconciling data against a database.

When dealing with connections to a database, there are two different options: SQL Server .NET Data Provider (System.Data.SqlClient) and OLE DB .NET Data Provider (System.Data.OleDb). In these samples we will use the SQL Server .NET Data Provider. These are written to talk directly to Microsoft SQL Server. The OLE DB .NET Data Provider is used to talk to any OLE DB provider (as it uses OLE DB underneath).

Connections :

Connections are used to 'talk to' databases, and are represented by provider-specific classes such as SQLConnection. Commands travel over connections and result sets are returned in the form of streams which can be read by a Data Reader object, or pushed into a Dataset object.

Commands

Commands contain the information that is submitted to a database, and are represented by provider-specific classes such as SQLCommand. A command can be a stored procedure call, an UPDATE statement, or a statement that returns results. You can also use input and output parameters, and return values as part of your command syntax. The example below shows how to issue an INSERT statement against the North wind database.

Data Readers

The Data Reader object is somewhat synonymous with a read-only/forward-only cursor over data. The Data Reader API supports flat as well as hierarchical data. A Data Reader object is returned after executing a command against a database. The format of the returned Data Reader object is different from a record set. For example, you might use the Data Reader to show the results of a search list in a web page.

Datasets

The Dataset object is similar to the ADO Record set object, but more powerful, and with one other important distinction: the Dataset is always disconnected. The Dataset object represents a cache of data, with database-like structures such as tables, columns, relationships, and constraints. However, though a Dataset can and does behave much like a database, it is important to remember that Dataset objects do not interact directly with databases, or other source data. This allows the developer to work with a programming model that is always consistent, regardless of where the source data resides. Data coming from a database, an XML file, from code, or user input can all be placed into

Dataset objects. Then, as changes are made to the Dataset they can be tracked and verified before updating the source data. The Get Changes method of the Dataset object actually creates a second Dataset that contains only the changes to the data. This Dataset is then used by a Data Adapter (or other objects) to update the original data source. The Dataset has many XML characteristics, including the ability to produce and consume XML data and XML schemas. XML schemas can be used to describe schemas interchanged via Web Services. In fact, a Dataset with a schema can actually be compiled for type safety and statement completion.

Data Adapters (OLEDB/SQL)

The Data Adapter object works as a bridge between the Dataset and the source data. Using the provider-specific SqlDataAdapter (along with its associated SqlCommand and SqlConnection) can increase overall performance when working with a Microsoft SQL Server databases. For other OLE DB-supported databases, you would use the OleDbDataAdapter object and its associated OleDbCommand and OleDbConnection objects. The Data Adapter object uses commands to update the data source after changes have been made to the Dataset. Using the Fill method of the Data Adapter calls the SELECT command; using the Update method calls the INSERT, UPDATE or DELETE command for each changed row. You can explicitly set these commands in order to control the statements used at runtime to resolve changes, including the use of stored procedures. For ad-hoc scenarios, a Command Builder object can generate these at run-time based upon a select statement. However, this run-time generation requires an extra round-trip to the server in order to gather required metadata, so explicitly providing the INSERT, UPDATE, and DELETE commands at design time will result in better run-time performance.

1. ADO.NET is the next evolution of ADO for the .Net Framework.
2. ADO.NET was created with n-Tier, statelessness and XML in the forefront. Two new objects, the Dataset and Data Adapter, are provided for these scenarios. ADO.NET can be used to get data from a stream, or to store data in a cache for updates.
3. There is a lot more information about ADO.NET in the documentation.

4. Remember, you can execute a command directly against the database in order to do inserts, updates, and deletes. You don't need to first put data into a Dataset in order to insert, update, or delete it.

5. Also, you can use a Dataset to bind to the data, move through the data, and navigate data relationships

Client-side Script(JAVASCRIPT):-

JavaScript:

JavaScript is a new scripting language for WebPages. Scripts written with java script can be embedded into your HTML pages. With java script you have many possibilities for enhancing your HTML page with interesting elements. For example you are able to respond to user-initiated events quite easily. Some effects that are now possible with java script were some time ago only possible with CGI. So you can create really sophisticated pages with the helps of java script on the Internet.

How can Java Script scripts run?

The first browser to support java script was the Netscape Navigator 2.0 of course the higher versions do have java script as well. You might know that java does not run on all Netscape Navigators 2.0 (or higher versions) versions. But this is not true for java script -although there are some problems with the different versions.

The Mac version for example seems to have many bugs. In the near future there are going to be some other browsers, which support java script. The Microsoft Internet explorer 3.0 is going to support java script. JavaScript enabled browsers are going to spread soon - it is worth learning

this new technique now. You might realize that is really easy to write Java Script scripts. We have to know is some basic techniques and some work-around for problems you might encounter. Of course we need a basic. Understanding HTML before reading this tutorial you can find many really good online resources about HTML. Best you make an online search about 'html' at yahoo if you want to get informed about HTML. Now I want to show some small scripts so you can learn how they are implemented into HTML-documents and to show which possibilities you have with the new scripting language. The following is a very small script, which will only print a text into an HTML document.

```
<html>
<head>
 My first JavaScript
</head>
<body><br>
This is a normal HTML document
<br>
<script language="JavaScript">
Document.write ("this is a java script")
</script><b r>
Backing HTML again
</body>
</html>
```

If you are using a java script enabled-browser at the moment then you will have the possibility to see this script working. If your browser doesn't support Java Script then this output might be some kind of strange…

This is a normal HTML document
This is java script!
Back in HTML again.

Functions

Functions are bet declared between the <Head> tag of HTML page. Functions are called by user-initiated events. Seems reasonable to keep the functions between the <Head> tags. They are loaded first before a user can do anything that might call a function. Scripts can be placed between inside comment fields to ensure that older browser do not display the script itself.

```
<html>
<head>
<script language=”JavaScript”>
 function pushbutton (){
     alert (“Hello!”);
}
</script>
</head>
<body>
<form>
<input type=”button” name=”Button1” value=”push me” onclick=”pushbutton ()”>
</form>
</body>
</html>
```

If we want to test this one immediately and you are using a Java Script enabled browser then please go ahead and push the button.

This script will create a button and when you press it a window will pop up saying “hello!”. In fact we have a lot of possibilities just by adding functions to our scripts.

The common browsers transmit the form information by either method: here’s the complete tag including the GET transmission method attribute for the previous form

Example

<Form method =GET action=http://www.mycompany.com/cgi-bin/upfdate.pl>
……
</form>

Input elements.

Use the <input> tag to define any one of a number of common form elements including text fields multiple choice lists click able images and submission buttons. There are many attributers for this tag only that types and name attributes are required for each element, each type of input element uses only a subset of the followed attributes. Additional <input> attributes may be required based upon which type of the form element you specify.

Submit button:

The submit button (<input type=submit>) does what its name implies, settings in motion the form's submission to the server from the browser. We many have more than submit buttons will be added to the parameter list the browser sends along to the server.

Example
< Input type =”submit”>
<Input type=”submit” value=”submit” name=”name”>

Reset button:

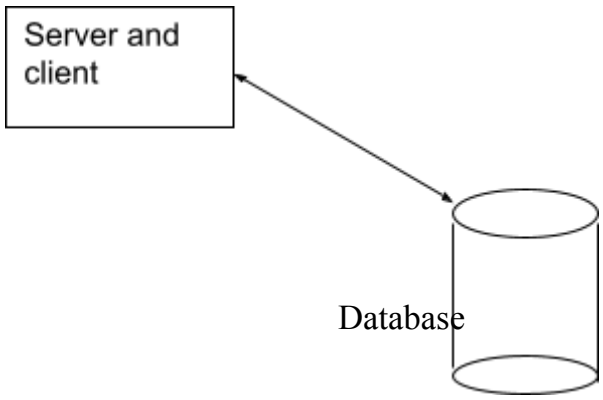
The reset button if firm <input> button is nearly self- explanatory; it lets the user reset erase or set to some default value all elements in the form. By default the browser displays a reset button worth the label “reset”. We can change that by specifying a value attribute with tour own button label.

DATABASE MODELS

ADO.NET and accessing the database through applets and ADO.NET API via an intermediate server resulted server resulted in a new type of database model which is different from the client-server model. Based on number of intermediate server through the request should go it is named as single tire, two tire and multi tire architecture

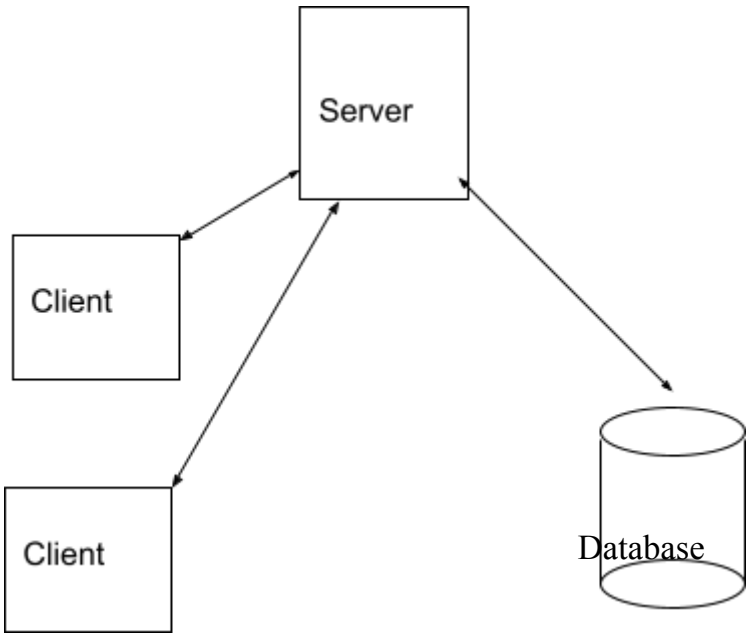
Single Tier

In a single tier the server and client are the same in the sense that a client program that needs information (client) and the source of this type of architecture is also possible in java, in case flat files are used to store the data. However this is useful only in case of small applications. The advantage with this is the simplicity and portability of the application developed.



Two Tier (client-server)

In two tier architecture the database resides in one machine and client in different machine they are connected through the network. In this type of architecture a database management takes control of the database and provides access to clients in a network. This software bundle is also called as the server. Software in different machines, requesting for information are called as the clients.



Three Tier and N-Tier

In the three-tier architecture, any number servers can access the database that resides on server. Which in turn serve clients in a network. For example, you want to access the database using java applets, the applet running in some other machine, can send request only to the server from which it is down loaded. For this reason we will need to have a intermediate server which will accept the requests from applets and them to the actual database server. This intermediate server acts as a two-way communication channel also. This is the information or data from the database is passed on to the applet that is requesting it. This can be extended to make n tiers of servers, each server carrying to specific type of request from clients, however in practice only 3 tiers architecture is popular.

C# Language

C# (pronounced C Sharp) is a multi-paradigm programming language that encompasses functional, imperative, generic, object-oriented (class-based), and component-oriented programming disciplines. It was developed by Microsoft as part of the .NET initiative and later approved as a standard by ECMA (**ECMA-334**) and ISO (**ISO/IEC 23270**). C# is one of the 44 programming languages supported by the .NET Framework's Common Language Runtime.

C# is intended to be a simple, modern, general-purpose, object-oriented programming language. Anders Hejlsberg, the designer of Delphi, leads the team which is developing C#. It has an object-oriented syntax based on C++ and is heavily influenced by other programming languages such as Delphi and Java. It was initially named Cool, which stood for "C like Object Oriented Language". However, in July 2000, when Microsoft made the project public, the name of the programming language was given as C#. The most recent version of the language is C# 3.0 which was released in conjunction with the .NET Framework 3.5 in 2007. The next proposed version, C# 4.0, is in development.

History:-

In 1996, Sun Microsystems released the Java programming language with Microsoft soon purchasing a license to implement it in their operating system. Java was originally meant to be a platform independent language, but Microsoft, in their implementation, broke their license agreement and made a few changes that would essentially inhibit Java's platform-independent capabilities. Sun filed a lawsuit and Microsoft settled, deciding to create their own version of a partially compiled, partially interpreted object-oriented programming language with syntax closely related to that of C++.

During the development of .NET, the class libraries were originally written in a language/compiler called Simple Managed C (SMC). In January 1999, Anders Hejlsberg formed a team to build a new language at the time called Cool, which stood for "C like Object Oriented Language".Microsoft had considered keeping the name "Cool" as the final name of the language, but chose not to do so for trademark reasons. By the time the .NET project was publicly announced at the July 2000 Professional Developers Conference, the language had been renamed C#, and the class libraries and ASP.NET runtime had been ported to C#.

C#'s principal designer and lead architect at Microsoft is Anders Hejlsberg, who was previously involved with the design of Visual J++, Borland Delphi, and Turbo Pascal. In interviews and technical papers he has stated that flaws in most major programming languages (e.g. C++, Java, Delphi, and Smalltalk) drove the fundamentals of the Common Language Runtime (CLR), which, in turn, drove the design of the C# programming language itself. Some argue that C# shares roots in other languages.

Features of C#:-

By design, C# is the programming language that most directly reflects the underlying Common Language Infrastructure (CLI). Most of C#'s intrinsic types correspond to value-types implemented by the CLI framework. However, the C# language specification does not state the code generation requirements of the compiler: that is, it does not state that a C# compiler must target a Common Language Runtime (CLR), or generate Common Intermediate Language (CIL), or generate any other specific format. Theoretically, a C# compiler could generate machine code like traditional compilers of C++ or FORTRAN; in practice, all existing C# implementations target CIL.

Some notable C# distinguishing features are:

- There are no global variables or functions. All methods and members must be declared within classes. It is possible, however, to use static methods/variables within public classes instead of global variables/functions.
- Local variables cannot shadow variables of the enclosing block, unlike C and C++. Variable shadowing is often considered confusing by C++ texts.
- C# supports a strict Boolean data type, `bool`. Statements that take conditions, such as `while` and `if`, require an expression of a boolean type. While C++ also has a boolean type, it can be freely converted to and from integers, and expressions such as `if(a)` require only that `a` is convertible to bool, allowing `a` to be an int, or a pointer. C# disallows this "integer meaning true or false" approach on the grounds that forcing programmers to use expressions that return exactly `bool` can prevent certain types of programming mistakes such as `if (a = b)` (use of = instead of ==).
- In C#, memory address pointers can only be used within blocks specifically marked as *unsafe*, and programs with unsafe code need appropriate permissions to run. Most object access is done through safe object references, which are always either pointing to a valid, existing object, or have the well-defined null value; a reference to a garbage-collected object, or to random block of memory, is impossible to obtain. An unsafe pointer can point to an instance of a value-type, array, string, or a block of memory allocated on a stack. Code that is not marked as unsafe can still store and manipulate pointers through the `System.IntPtr` type, but cannot dereference them.

- Managed memory cannot be explicitly freed, but is automatically garbage collected. Garbage collection addresses memory leaks. C# also provides direct support for deterministic finalization with the `using` statement (supporting the Resource Acquisition Is Initialization idiom).
- Multiple inheritance is not supported, although a class can implement any number of interfaces. This was a design decision by the language's lead architect to avoid complication, avoid dependency hell and simplify architectural requirements throughout CLI.
- C# is more type safe than C++. The only implicit conversions by default are those which are considered safe, such as widening of integers and conversion from a derived type to a base type. This is enforced at compile-time, during JIT, and, in some cases, at runtime. There are no implicit conversions between booleans and integers, nor between enumeration members and integers (except for literal 0, which can be implicitly converted to any enumerated type). Any user-defined conversion must be explicitly marked as explicit or implicit, unlike C++ copy constructors (which are implicit by default) and conversion operators (which are always implicit).
- Enumeration members are placed in their own scope.
- C# provides syntactic sugar for a common pattern of a pair of methods, accessor (getter) and mutator (setter) encapsulating operations on a single attribute of a class, in form of properties.
- Full type reflection and discovery is available.
- C# currently (as of 3 June 2008) has 77 reserved words.

Common Type system (CTS)

C# has a *unified type system*. This unified type system is called **Common Type System (CTS)**.

A unified type system implies that all types, including primitives such as integers, are subclasses of the `System.Object` class. For example, every type inherits a `ToString()` method. For performance reasons, primitive types (and value types in general) are internally allocated on the stack.

Categories of datatypes

CTS separates datatypes into two categories:

- Value types
- Reference types

Value types are plain aggregations of data. Instances of value types do not have referential identity nor a referential comparison semantics - equality and inequality comparisons for value types compare the actual data values within the instances, unless the corresponding operators are overloaded. Value types are derived from `System.ValueType`, always have a default value, and can always be created and copied. Some other limitations on value types are that they cannot derive from each other (but can implement interfaces) and cannot have a default (parameterless) constructor. Examples of value types are some primitive types, such as `int` (a signed 32-bit integer), `float` (a 32-bit IEEE floating-point number), `char` (a 16-bit Unicode codepoint), and `System.DateTime` (identifies a specific point in time with millisecond precision).

In contrast, reference types have the notion of referential identity - each instance of reference type is inherently distinct from every other instance, even if the data within both instances is the same. This is reflected in default equality and inequality comparisons for reference types, which test for referential rather than structural equality, unless the corresponding operators are overloaded (such as the case for `System.String`). In general, it is not always possible to create an instance of a reference

type, nor to copy an existing instance, or perform a value comparison on two existing instances, though specific reference types can provide such services by exposing a public constructor or implementing a corresponding interface (such as `ICloneable` or `IComparable`). Examples of reference types are `object` (the ultimate base class for all other C# classes), `System.String` (a string of Unicode characters), and `System.Array` (a base class for all C# arrays).

Both type categories are extensible with user-defined types.

Boxing and unboxing

Boxing is the operation of converting a value of a value type into a value of a corresponding reference type.

Example:

```
int foo = 42;         // Value type...
object bar = foo;     // foo is boxed to bar.
```

Unboxing is the operation of converting a value of a reference type (previously boxed) into a value of a value type.

Example:

```
int foo = 42;         // Value type.
object bar = foo;     // foo is boxed to bar.
int foo2 = (int)bar;  // Unboxed back to value type.
```

Features of C# 2.0

New features in C# for the .NET SDK 2.0 (corresponding to the 3rd edition of the ECMA-334 standard) are:

Partial class

Partial classes allow implementation of a class to be spread between several files, with each file containing one or more class members. It is primary useful when parts of a class are automatically generated. For example, the feature is heavily used by code-generating user interface designers in Visual Studio.

file1.cs:

```
public partial class MyClass
{
    public void MyMethod1()
    {
        // Manually written code
    }
}
```

file2.cs:

```
public partial class MyClass
{
    public void MyMethod2()
    {
        // Automatically generated code
    }
}
```

Generics

Generics, or parameterized types, or parametric polymorphism is a .NET 2.0 feature supported by C#. Unlike C++ templates, .NET parameterized types are instantiated at runtime rather than by the compiler; hence they can be cross-language whereas C++ templates cannot. They support some features not supported directly by C++ templates such as type constraints on generic parameters by use of interfaces. On the other hand, C# does not support non-type generic parameters. Unlike

generics in Java, .NET generics use reification to make parameterized types first-class objects in the CLI Virtual Machine, which allows for optimizations and preservation of the type information.

Static classes

Static classes are classes that cannot be instantiated or inherited from, and that only allow static members. Their purpose is similar to that of modules in many procedural languages.

A new form of iterator providing generator functionality

A new form of iterator that provides generator functionality, using a `yield return` construct similar to `yield` in Python.

```
// Method that takes an iterable input (possibly an array)
// and returns all even numbers.
public static IEnumerable<int> GetEven(IEnumerable<int> numbers)
{
    foreach (int i in numbers)
    {
        if (i % 2 == 0) yield return i;
    }
}
```

Anonymous delegates

Anonymous delegates provide closure functionality in C#. Code inside the body of an anonymous delegate has full read/write access to local variables, method parameters, and class members in scope of the delegate, excepting `out` and `ref` parameters. For example:-

```
int SumOfArrayElements(int[] array)
{
    int sum = 0;
    Array.ForEach(
        array,
```

```
            delegate(int x)
            {
                sum += x;
            }
        );
        return sum;
    }
```

Delegate covariance and contravariance

Conversions from method groups to delegate types are covariant and contravariant in return and parameter types, respectively.

The accessibility of property accessors can be set independently

Example:

```
string status = string.Empty;

public string Status
{
    get { return status; }              // anyone can get value of
this property,
    protected set { status = value; }  // but only derived classes
can change it
}
```

Nullable types

Nullable value types (denoted by a question mark, e.g. `int? i = null;`) which add `null` to the set of allowed values for any value type. This provides improved interaction with SQL databases, which can have nullable columns of types corresponding to C# primitive types: an SQL `INTEGER NULL` column type directly translates to the C# `int?`.

Nullable types received an eleventh-hour improvement at the end of August 2005, mere weeks before the official launch, to improve their boxing characteristics: a nullable variable which is assigned null is not actually a null reference, but rather an instance of `struct Nullable<T>` with property `HasValue` equal to `false`. When boxed, the `Nullable` instance itself is boxed, and not the value stored in it, so the resulting reference would always be non-null, even for null values. The following code illustrates the corrected flaw:

```
int? i = null;
object o = i;
if (o == null)
    Console.WriteLine("Correct behaviour - runtime version from
September 2005 or later");
else
    Console.WriteLine("Incorrect behaviour - pre-release runtime
(from before September 2005)");
```

When copied into objects, the official release boxes values from `Nullable` instances, so null values and null references are considered equal. The late nature of this fix caused some controversy , since it required core-CLR changes affecting not only .NET2, but all dependent technologies (including C#, VB, SQL Server 2005 and Visual Studio 2005).

DATABASE TABLES:

Accidents Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
instype	varchar(10)	☐
inspector	varchar(15)	☐
acctype	varchar(10)	☑
cause	varchar(30)	☑
injured	int	☑
dead	int	☑
remarks	varchar(30)	☐

Annual Returns Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
product	varchar(10)	☐
capital	int	☐
workersnum	int	☐
profit	int	☑
loss	int	☑
tax	int	☑

Closed Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
doc	varchar(10)	☐
dor	varchar(10)	☑

Committee Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
name	varchar(20)	☑
usertype	varchar(20)	☑
address	varchar(100)	☑
contactno	varchar(20)	☑

Convictions Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
reason	varchar(20)	☐
trail	int	☑

Emp Details Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
numofemps	bigint	☑
totsalofemps	bigint	☑
bonusforemps	bigint	☑

Financial Details Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
Investment	int	☐
workersnum	int	☐
profit	int	☑
loss	int	☑
tax	int	☑

Partner Details Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
name	varchar(20)	☑
share	varchar(10)	☑
address	varchar(100)	☑
contactno	varchar(20)	☑

Productor Details Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☑
productid	int	☑
productname	varchar(20)	☑
quantity	varchar(20)	☑
cost	bigint	☑
orderdate	datetime	☑
compdate	datetime	☑

Registration Table

Column Name	Data Type	Allow Nulls
fname	varchar(30)	☐
fcode	varchar(10)	☐
ftype	varchar(15)	☐
fprocess	varchar(15)	☐
workersnum	int	☐
area	varchar(5)	☐
dor	varchar(10)	☐
hospital	varchar(5)	☐
canteen	varchar(5)	☐
ambulance	varchar(5)	☐
doctersnum	int	☐
address	varchar(30)	☐
city	varchar(15)	☐

FEASIBILITY STUDY:

Feasibility study is conducted once the problem is clearly understood. Feasibility study is a high level capsule version of the entire system analysis and design process. The objective is to determine quickly at a minimum expense how to solve a problem. The purpose of feasibility is not to solve the problem but to determine if the problem is worth solving.

The system has been tested for feasibility in the following points.

1. Technical Feasibility
2. Economical Feasibility
3. Operational Feasibility.

1. Technical Feasibility

The project entitles "Courier Service System" is technically feasibility because of the below mentioned feature. The project was developed in Java which Graphical User Interface.

It provides the high level of reliability, availability and compatibility. All these make Java an appropriate language for this project. Thus the existing software Java is a powerful language.

2. Economical Feasibility

The computerized system will help in automate the selection leading the profits and details of the organization. With this software, the machine and manpower utilization are expected to go up by 80-90% approximately. The costs incurred of not creating the system are set to be great, because precious time can be wanted by manually.

3. Operational Feasibility

In this project, the management will know the details of each project where he may be presented and the data will be maintained as decentralized and if any inquires for that particular contract can be known as per their requirements and necessaries.

Implementation:

Implementation is the stage where the theoretical design is turned into a working system. The most crucial stage in achieving a new successful system and in giving confidence on the new system for the users that it will work efficiently and effectively.

The system can be implemented only after thorough testing is done and if it is found to work according to the specification.

It involves careful planning, investigation of the current system and its constraints on implementation, design of methods to achieve the change over and an evaluation of change over methods a part from planning. Two major tasks of preparing the implementation are education and training of the users and testing of the system.

The more complex the system being implemented, the more involved will be the systems analysis and design effort required just for implementation.

The implementation phase comprises of several activities. The required hardware and software acquisition is carried out. The system may require some software to be developed. For this, programs

are written and tested. The user then changes over to his new fully tested system and the old system is discontinue

TESTING:

The testing phase is an important part of software development. It is the puterized system will help in automate process of finding errors and missing operations and also a complete verification to determine whether the objectives are met and the user requirements are satisfied.

Software testing is carried out in three steps:

1. The first includes unit testing, where in each module is tested to provide its correctness, validity and also determine any missing operations and to verify whether the objectives have been met. Errors are noted down and corrected immediately. Unit testing is the important and major part of the project. So errors are rectified easily in particular module and program clarity is increased. In this project entire system is divided into several modules and is developed individually. So unit testing is conducted to individual modules.

2. The second step includes Integration testing. It need not be the case, the software whose modules when run individually and showing perfect results, will also show perfect results when run as a whole. The individual modules are clipped under this major module and tested again and verified the results. This is due to poor interfacing, which may results in data being lost across an interface. A module can have inadvertent, adverse effect on any other or on the global data structures, causing serious problems.

3.The final step involves validation and testing which determines which the software functions as the user expected. Here also some modifications were. In the completion of the project it is satisfied fully by the end user.

Maintenance and environment:

AS the number of computer based systems, grieve libraries of computer software began to expand. In house developed projects produced tones of thousand soft program source statements. Software products purchased from the outside added hundreds of thousands of new statements. A dark cloud appeared on the horizon. All of these programs, all of those source statements-had to be corrected when false were detected, modified as user requirements changed, or adapted to new hardware that was purchased. These activities were collectively called software Maintenance.

The maintenance phase focuses on change that is associated with error correction, adaptations required as the software's environment evolves, and changes due to enhancements brought about by changing customer requirements. Four types of changes are encountered during the maintenance phase.

Correction
Adaptation
Enhancement
Prevention

Correction:

Even with the best quality assurance activities is lightly that the customer will uncover defects in the software. Corrective maintenance changes the software to correct defects.

Maintenance is a set of software Engineering activities that occur after software has been delivered to the customer and put into operation. Software configuration management is a set of tracking and control activities that began when a software project begins and terminates only when the software is taken out of the operation.

We may define maintenance by describing four activities that are undertaken after a program is released for use

Corrective Maintenance

Adaptive Maintenance

Perfective Maintenance or Enhancement

Preventive Maintenance or reengineering

Only about 20 percent of all maintenance work are spent "fixing mistakes". The remaining 80 percent are spent adapting existing systems to changes in their external environment, making enhancements requested by users, and reengineering an application for use.

ADAPTATION:

Over time, the original environment (E>G., CPU, operating system, business rules, external product characteristics) for which the software was developed is likely to change. Adaptive maintenance results in modification to the software to accommodate change to its external environment.

ENHANCEMENT:

As software is used, the customer/user will recognize additional functions that will provide benefit. Perceptive maintenance extends the software beyond its original function requirements.

PREVENTION :

Computer software deteriorates due to change, and because of this, preventive maintenance, often called software re engineering, must be conducted to enable the software to serve the needs of its end

users. In essence, preventive maintenance makes changes to computer programs so that they can be more easily corrected, adapted, and enhanced. Software configuration management (SCM) is an umbrella activity that is applied throughout the software process. SCM activities are developed to

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

FSS

- 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.

CODE

Closed & Remove Factories

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class Closed_and_Removed_Factories : System.Web.UI.Page
{
    private SqlConnection cnn=new
SqlConnection("server=dataserver;database=factorystatistics;uid=sa;passwo
rd=sqlserver");
    protected void Page_Load(object sender, EventArgs e)
    {
        cnn.Open();
        SqlCommand cmd = new SqlCommand("select fname,fcode from
       Registeration ",cnn);
        SqlDataReader dr = cmd.ExecuteReader();
        dr.Read();
        DropDownList1.Items.Add(dr[0].ToString());
        DropDownList2.Items.Add(dr[1].ToString());
        dr.Close();
        cnn.Close();
    }
    protected void Button1_Click(object sender, EventArgs e)
    {
        cnn.Open();
        SqlCommand cmd1 = new SqlCommand("insert into
closed(fname,fcode,doc,dor)
```

```
values('"+DropDownList1.SelectedItem.ToString()+"','"+DropDownList2.Selec
tedItem.ToString()+"','"+TextBox1.Text+"','"+TextBox2.Text+"')",cnn);
        cmd1.ExecuteNonQuery();
        Response.Redirect("Closed and Removed Factories.aspx");
    }
}
```

Annual Returns:

```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class Annul_Return_Details : System.Web.UI.Page
{
    private SqlConnection cnn = new
SqlConnection("server=dataserver;database=factorystatistics;uid=sa;passwo
rd=sqlserver");
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            cnn.Open();
            SqlCommand cmd = new SqlCommand("select fname,fcode from
annulreturns ", cnn);
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                DropDownList1.Items.Add(dr[0].ToString());
                DropDownList2.Items.Add(dr[1].ToString());
            }
            dr.Close();
            cnn.Close();
        }
    }
    protected void DropDownList1_SelectedIndexChanged(object sender,
EventArgs e)
    {

    }
    protected void DropDownList2_SelectedIndexChanged(object sender,
EventArgs e)
```

```
    {
        ////       keep Auto PostBack = true
        cnn.Open();
        SqlDataAdapter da = new SqlDataAdapter("select
fname,fcode,product,capital,workersnum,profit,loss,tax from annulreturns
where fname='" + DropDownList1.SelectedItem.ToString() + "' and fcode ='"
+ DropDownList2.SelectedItem.ToString() + "'", cnn);
        DataSet ds = new DataSet();
        da.Fill(ds);
        GridView1.DataSource = ds;
        GridView1.DataBind();
    }
}
```

Conviction Details:

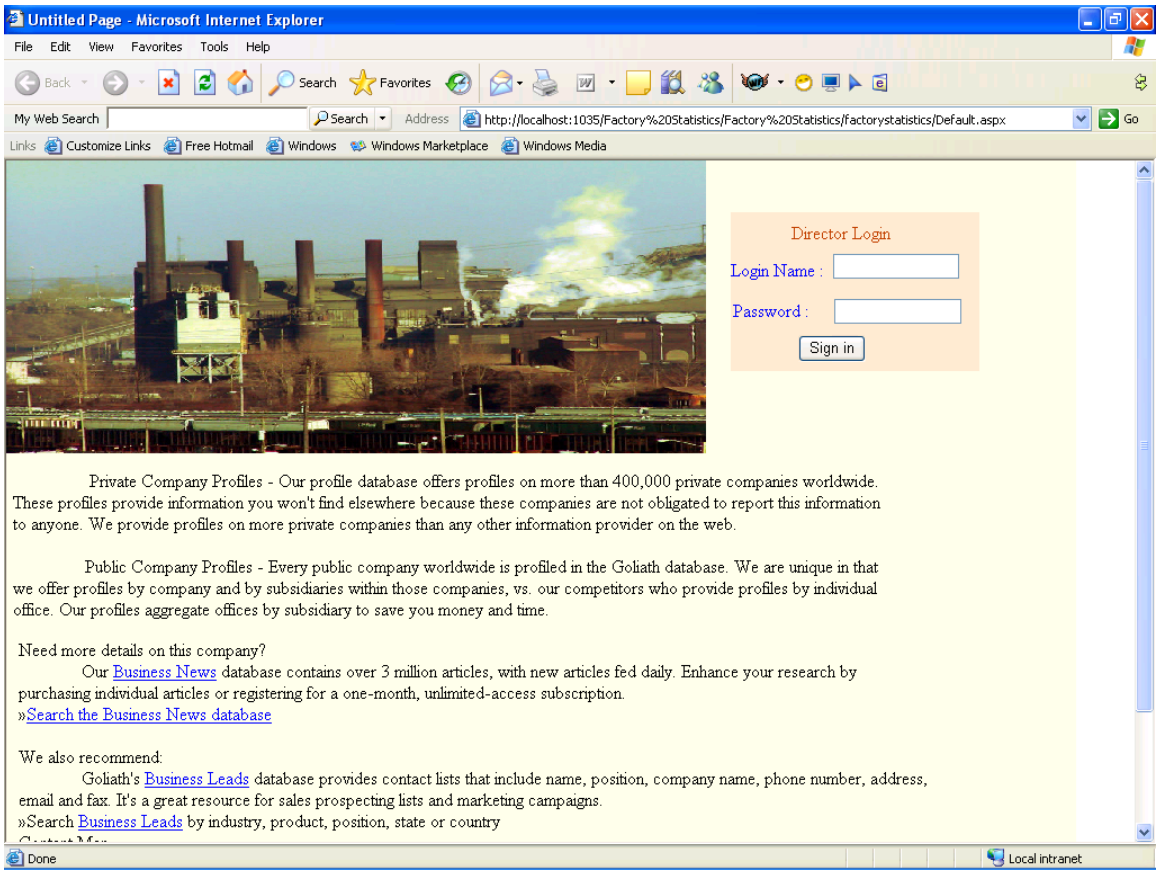
```
using System;
using System.Data;
using System.Configuration;
using System.Collections;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Web.UI.HtmlControls;
using System.Data.SqlClient;

public partial class Conviction_details : System.Web.UI.Page
{
    private SqlConnection cnn = new
SqlConnection("server=dataserver;database=factorystatistics;uid=sa;passwo
rd=sqlserver");
    protected void Page_Load(object sender, EventArgs e)
    {
        if (!IsPostBack)
        {
            cnn.Open();
            SqlCommand cmd = new SqlCommand("select fname,fcode from
convictions ", cnn);
            SqlDataReader dr = cmd.ExecuteReader();
            while (dr.Read())
            {
                DropDownList1.Items.Add(dr[0].ToString());
                DropDownList2.Items.Add(dr[1].ToString());
            }
            dr.Close();
            cnn.Close();
        }
    }
    protected void DropDownList2_SelectedIndexChanged(object sender,
EventArgs e)
    {

        ////       keep Auto PostBack = true
```

```
        cnn.Open();
        SqlDataAdapter da = new SqlDataAdapter("select
fname,fcode,reason,trail from convictions where
fname='"+DropDownList1.SelectedItem.ToString()+"' and fcode ='" +
DropDownList2.SelectedItem.ToString() + "'", cnn);
        DataSet ds = new DataSet();
        da.Fill(ds);
        GridView1.DataSource = ds;
        GridView1.DataBind();
    }
}
```

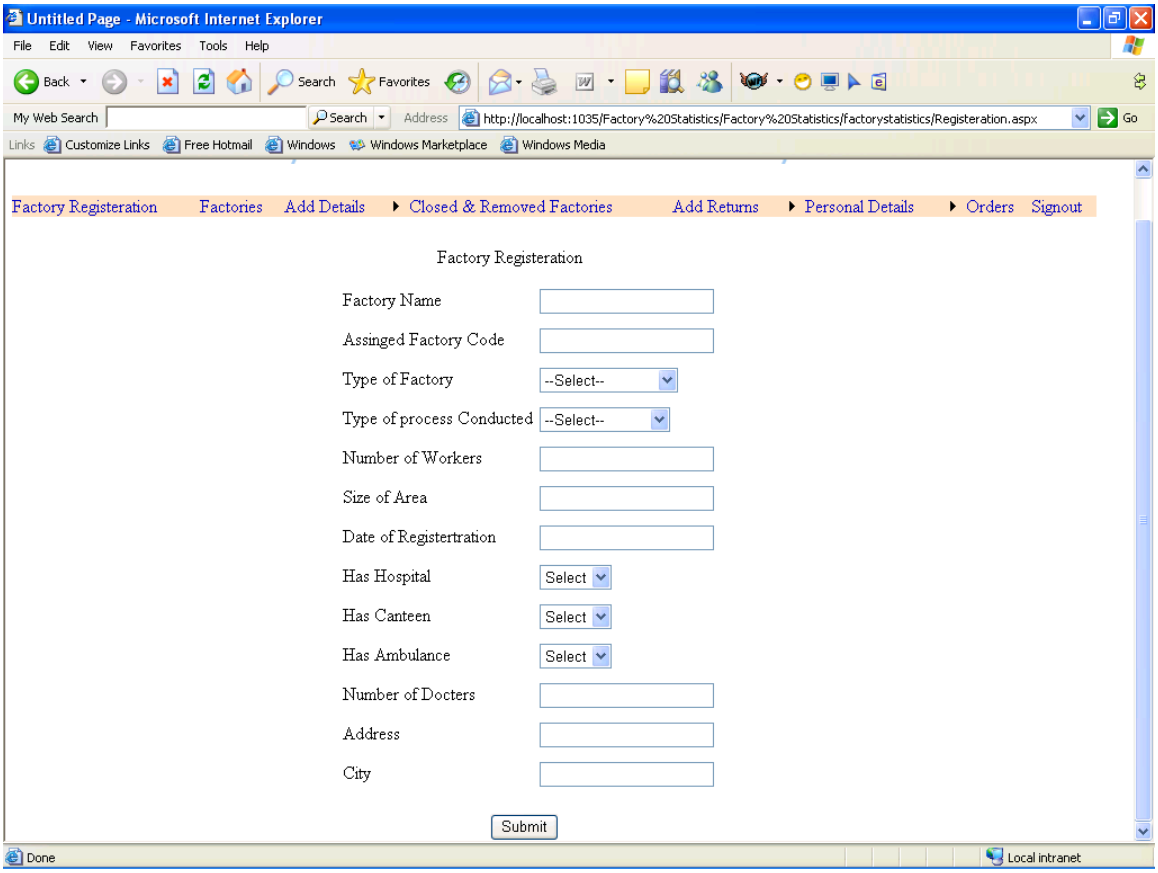
Output screens


Untitled Page - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites
My Web Search Search Address http://localhost:1035/Factory%20Statistics/Factory%20Statistics/factorystatistics/Default.aspx Go
Links Customize Links Free Hotmail Windows Windows Marketplace Windows Media
Director Login
Login Name :
Password :
Sign in
Private Company Profiles - Our profile database offers profiles on more than 400,000 private companies worldwide. These profiles provide information you won't find elsewhere because these companies are not obligated to report this information to anyone. We provide profiles on more private companies than any other information provider on the web.
Public Company Profiles - Every public company worldwide is profiled in the Goliath database. We are unique in that we offer profiles by company and by subsidiaries within those companies, vs. our competitors who provide profiles by individual office. Our profiles aggregate offices by subsidiary to save you money and time.
Need more details on this company?
Our Business News database contains over 3 million articles, with new articles fed daily. Enhance your research by purchasing individual articles or registering for a one-month, unlimited-access subscription.
»Search the Business News database
We also recommend:
Goliath's Business Leads database provides contact lists that include name, position, company name, phone number, address, email and fax. It's a great resource for sales prospecting lists and marketing campaigns.
»Search Business Leads by industry, product, position, state or country
Done
Local intranet

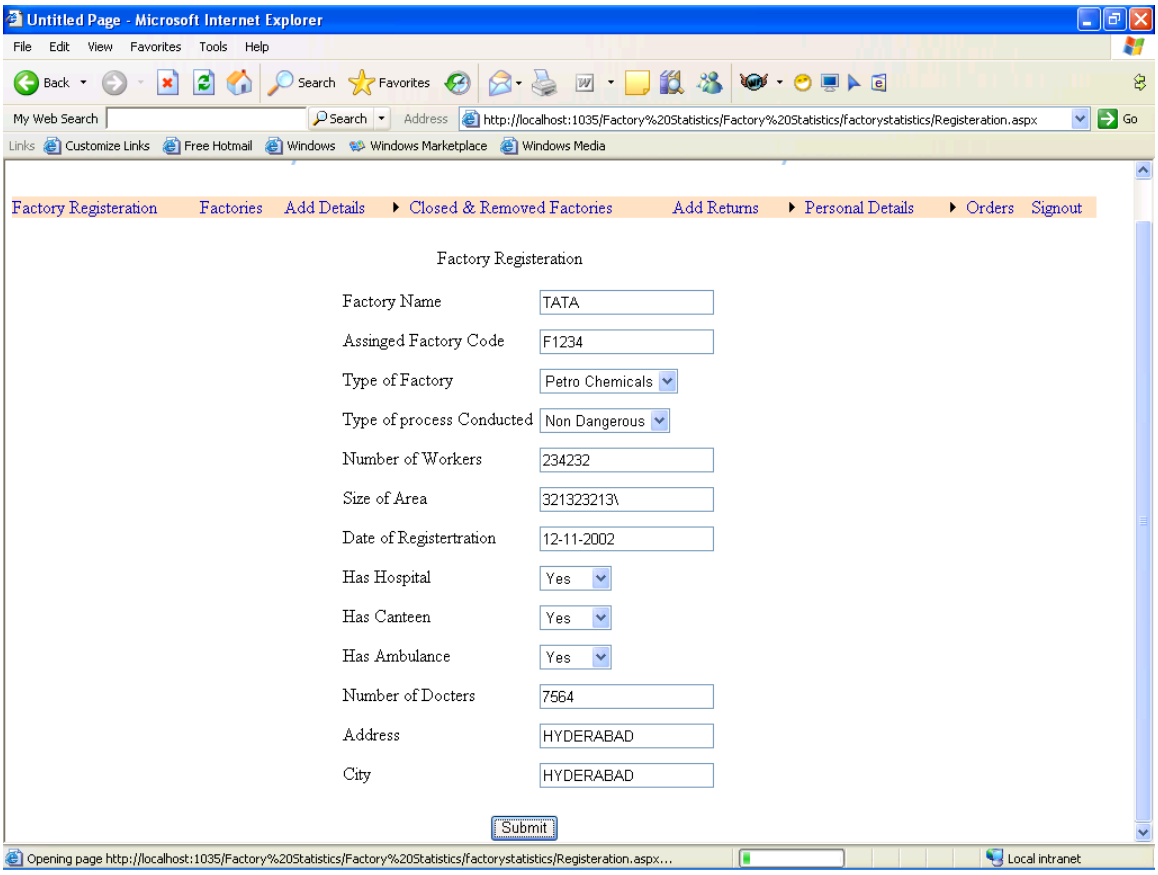
FSS

HOME PAGE

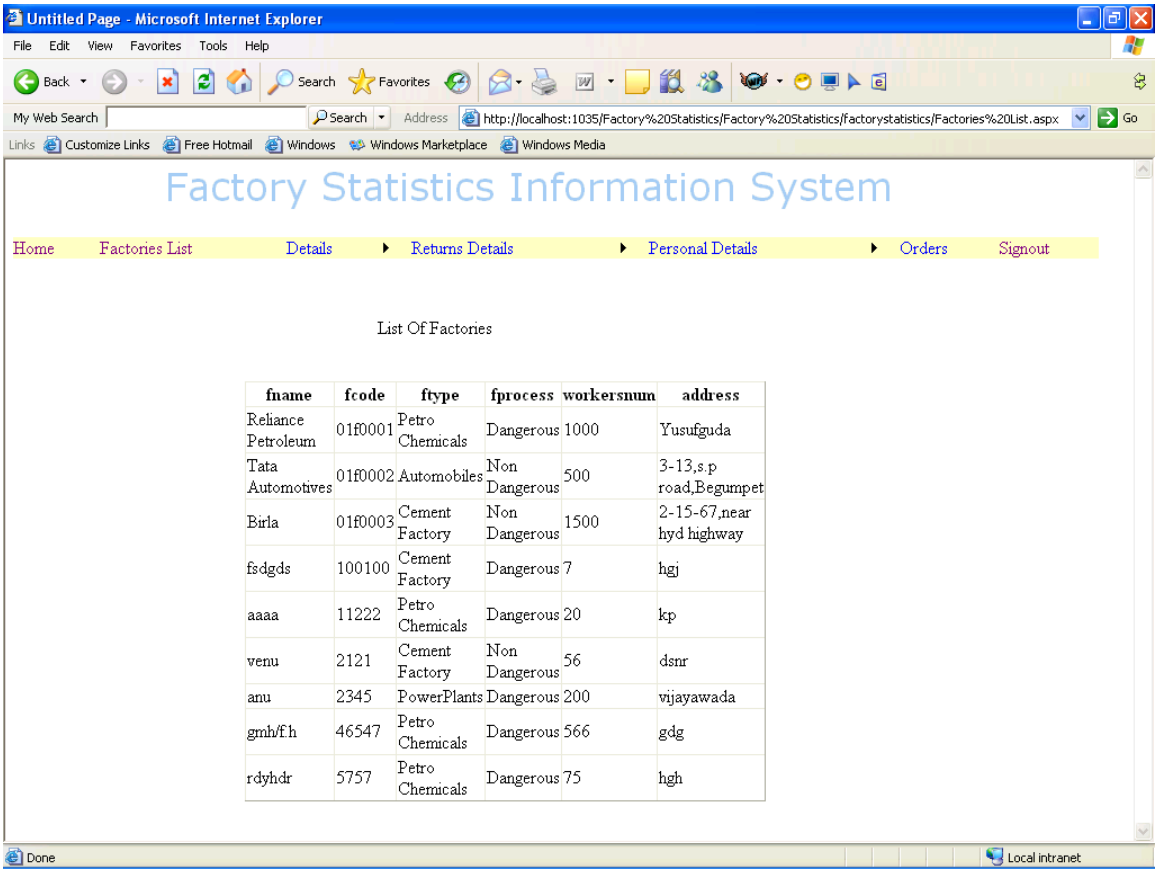
ADMIN LOGIN



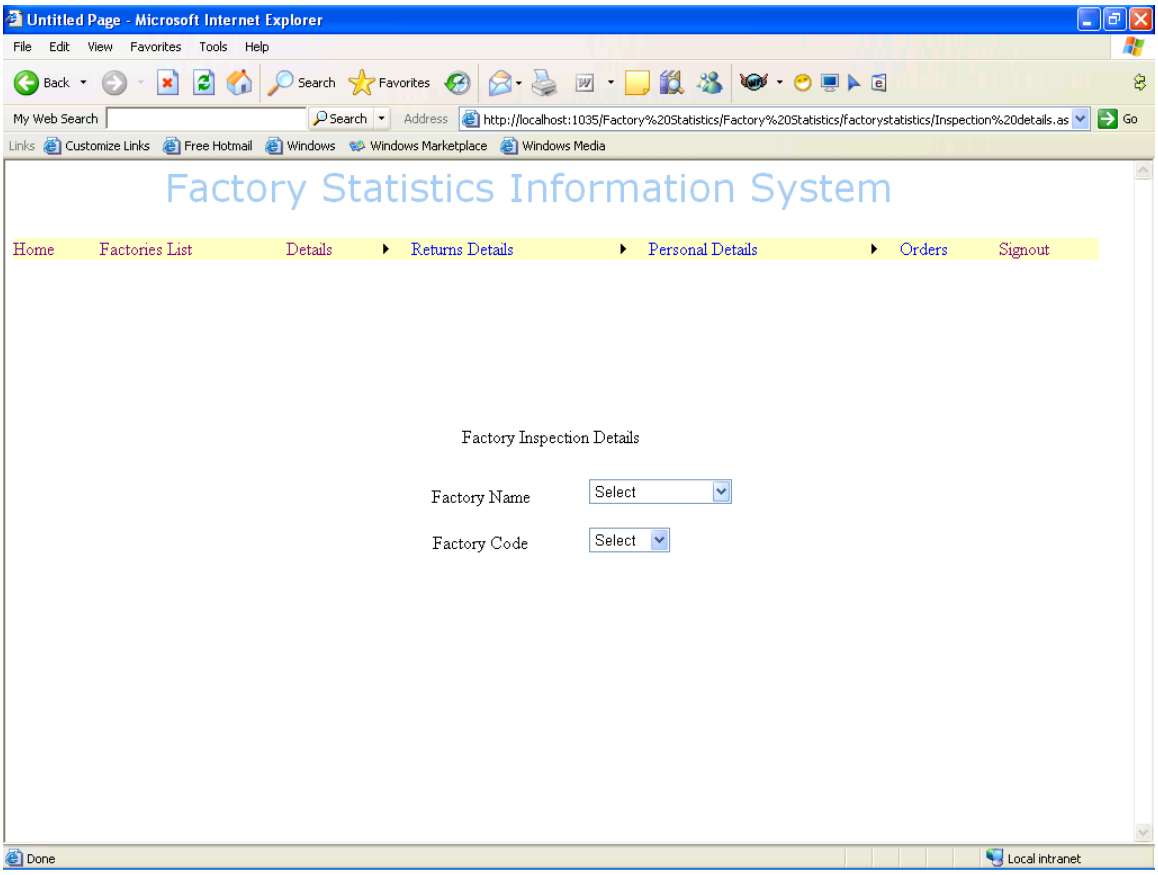
FACTORY REGISTRATION



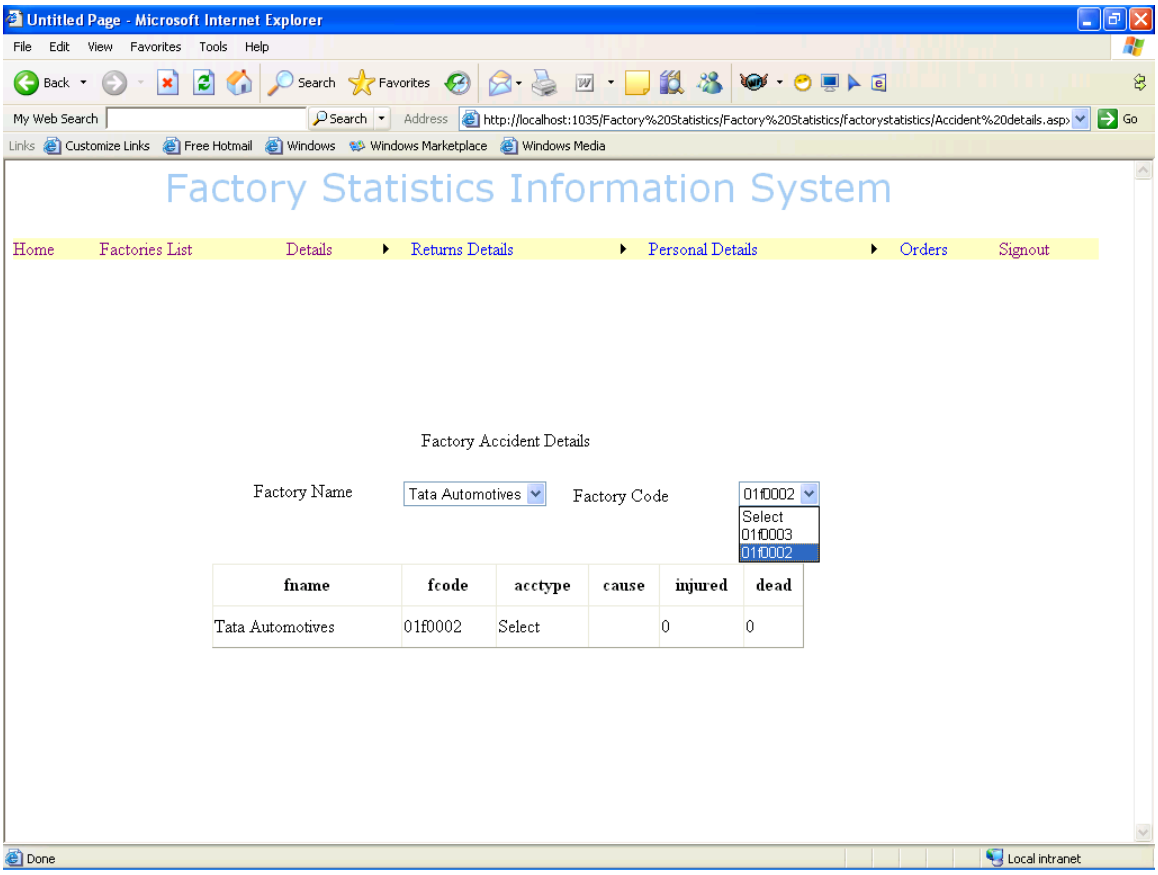
ENTERING VALUES



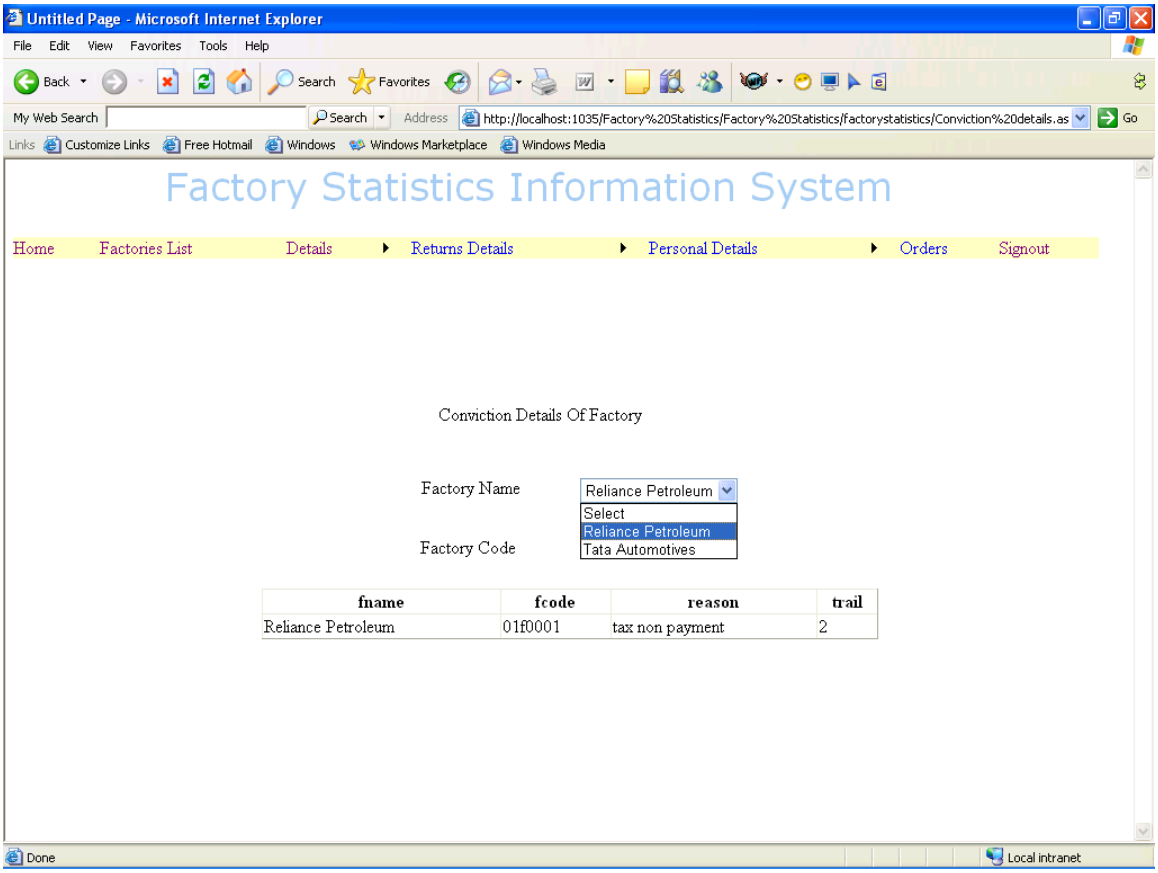
LIST OF FACTORIES



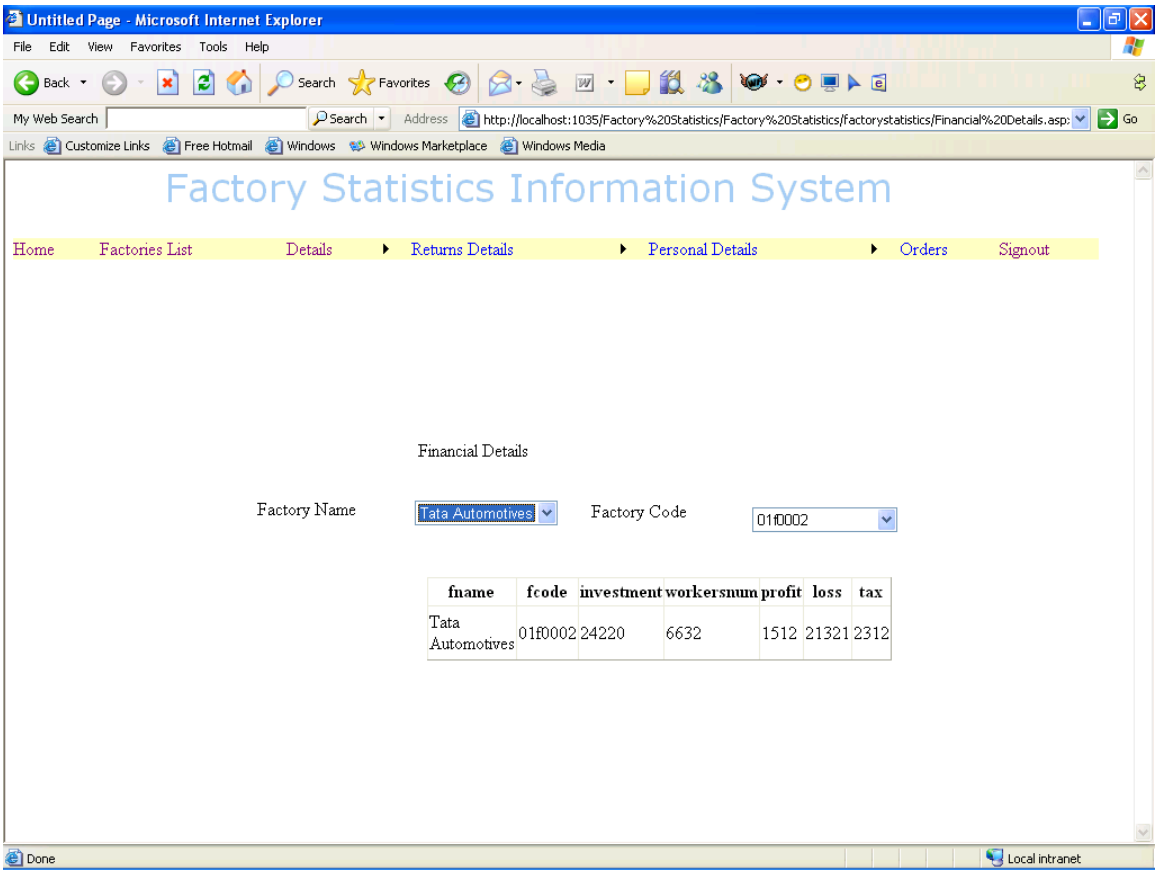
INSPECTION DETAILS



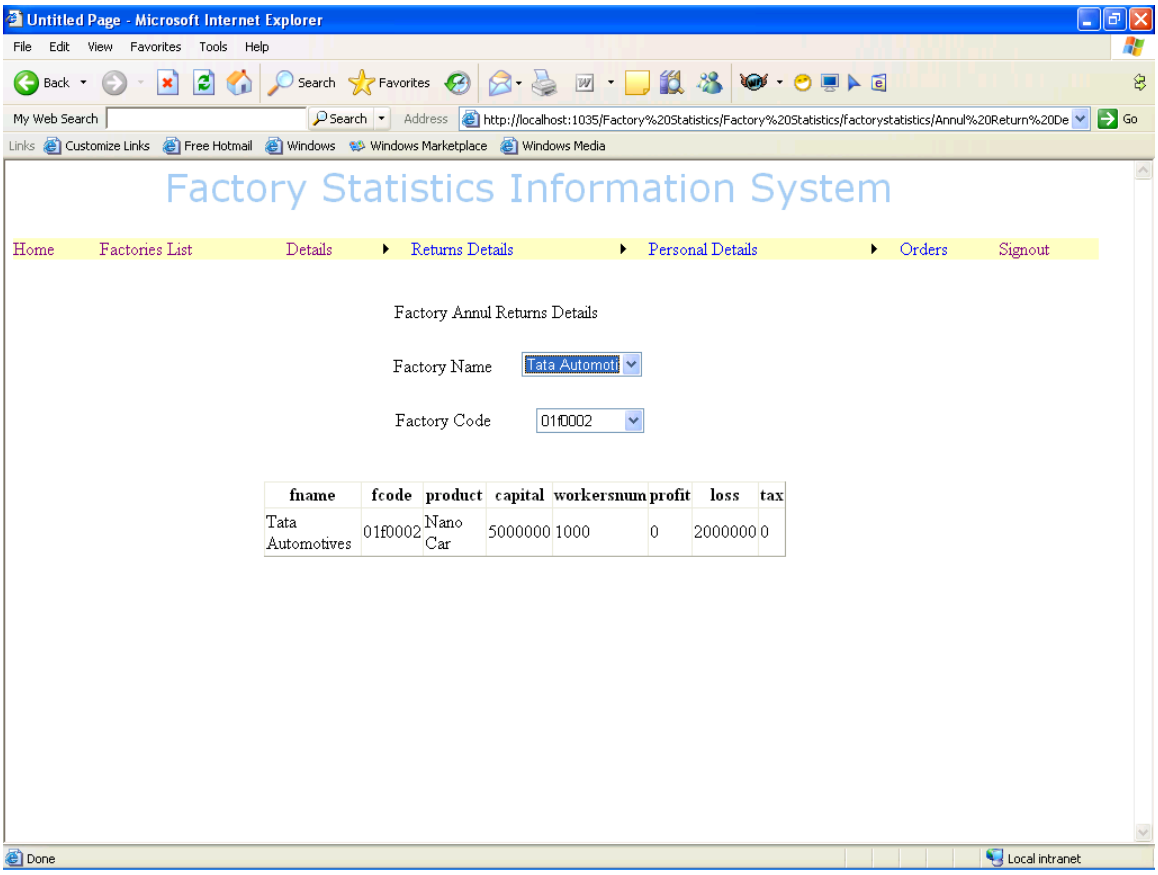
ACCIDENT DETAILS



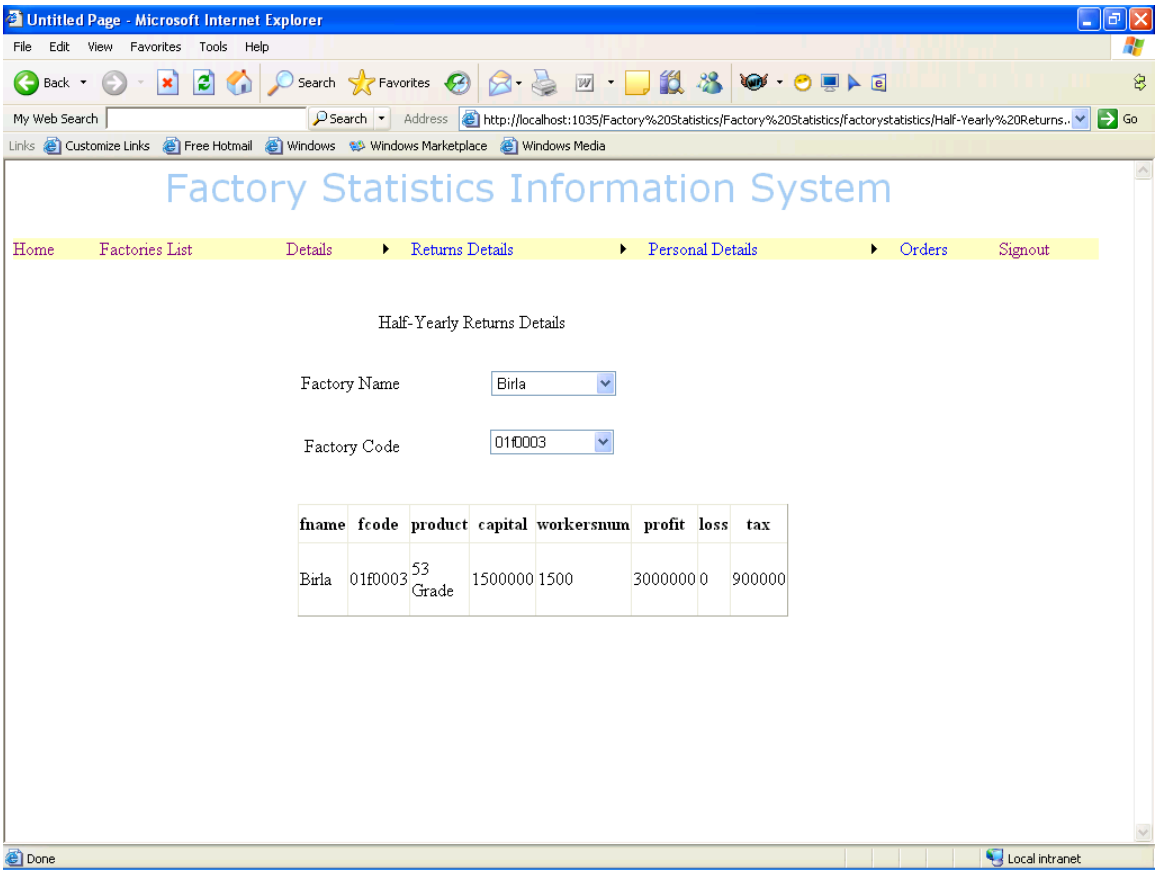
CONVICTION DETAILS



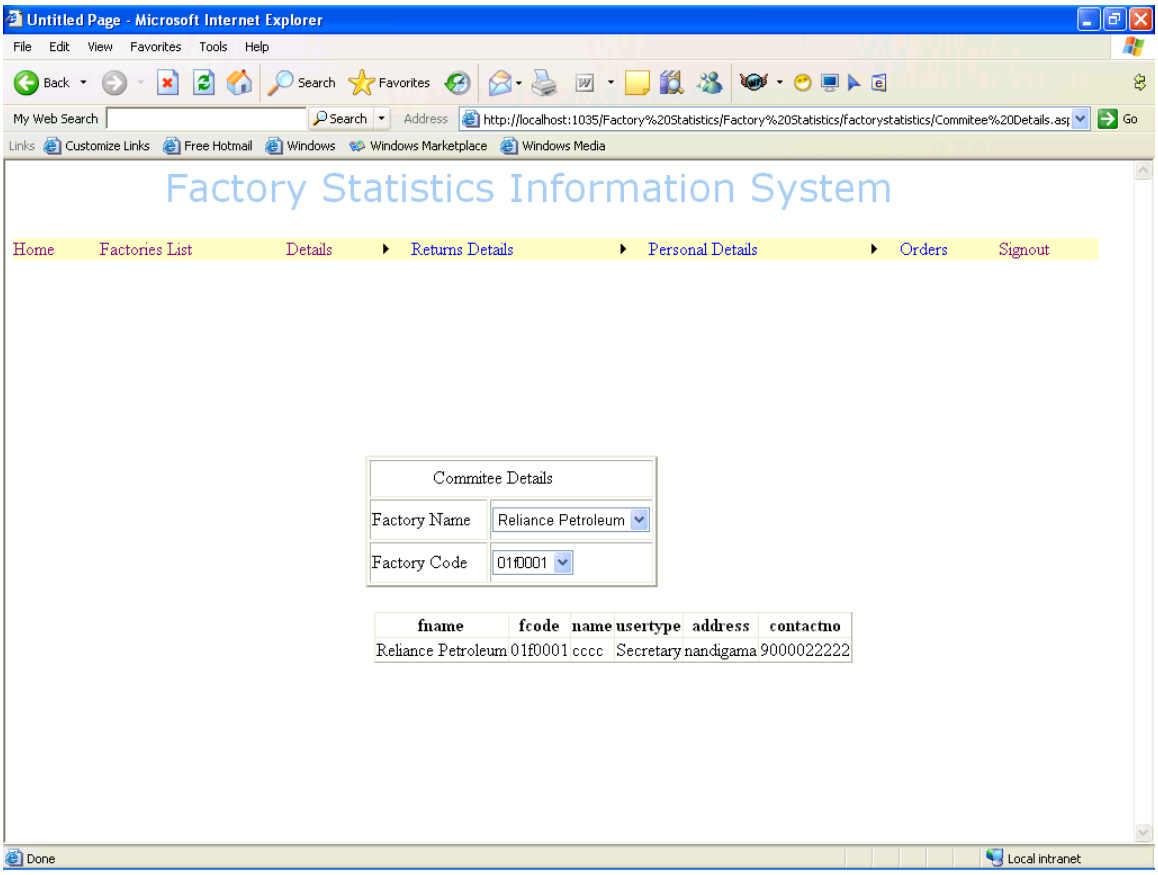
FINANCIAL DETAILS



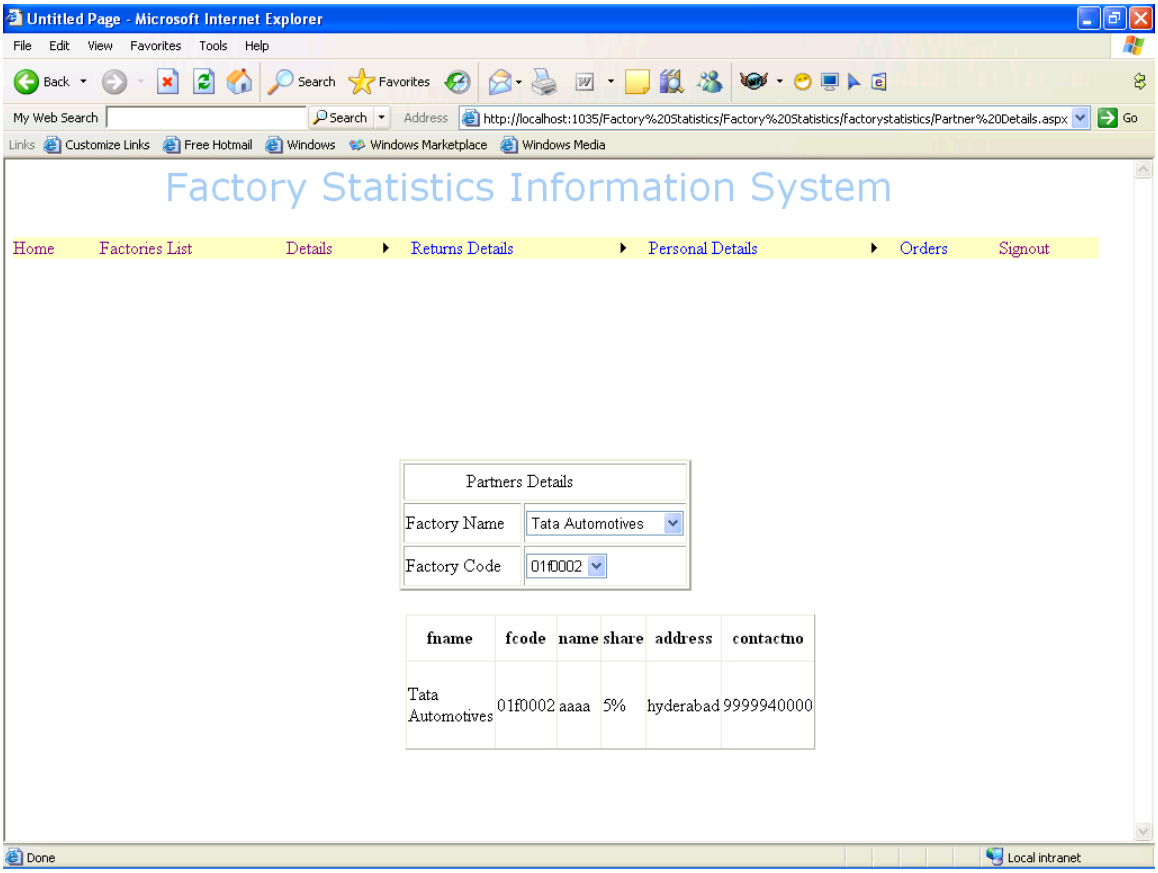
ANNUAL RETURNS DETAILS



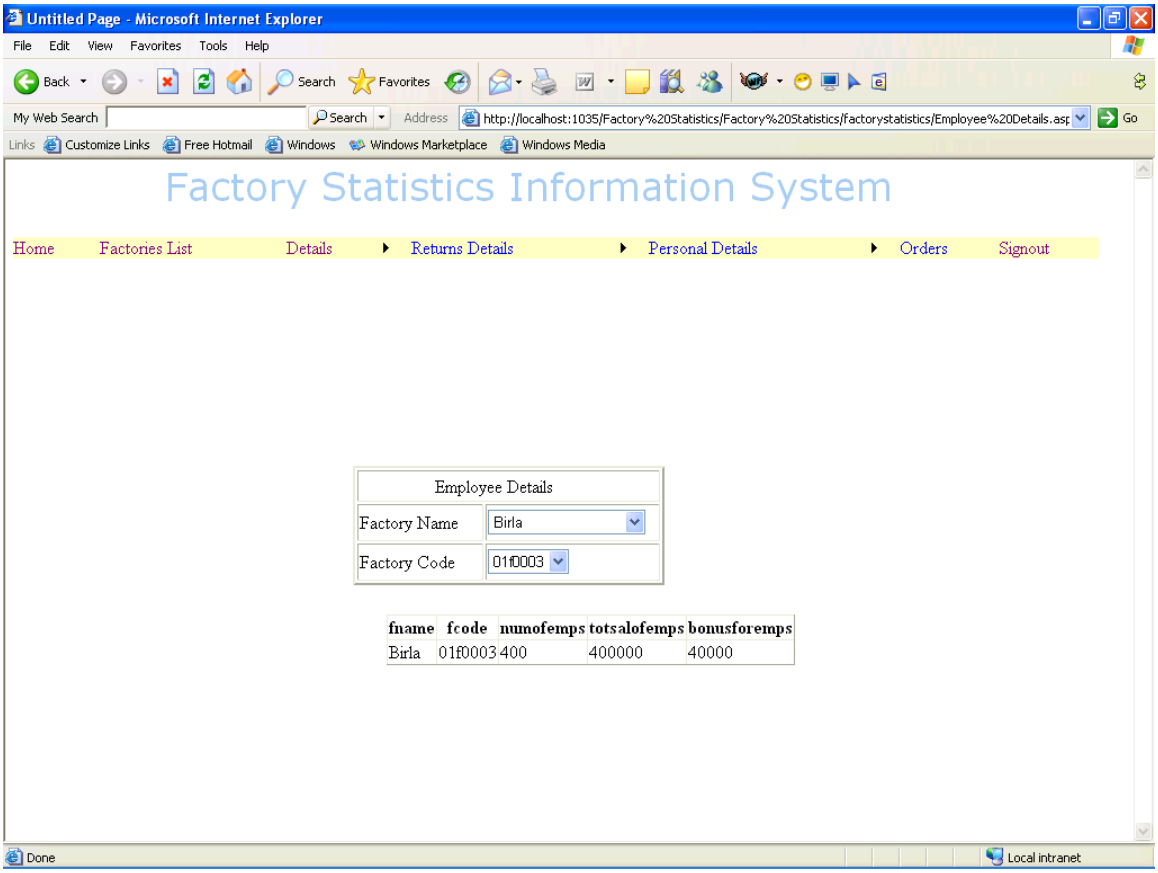
HALF YEARLY RETURNS DETAILS



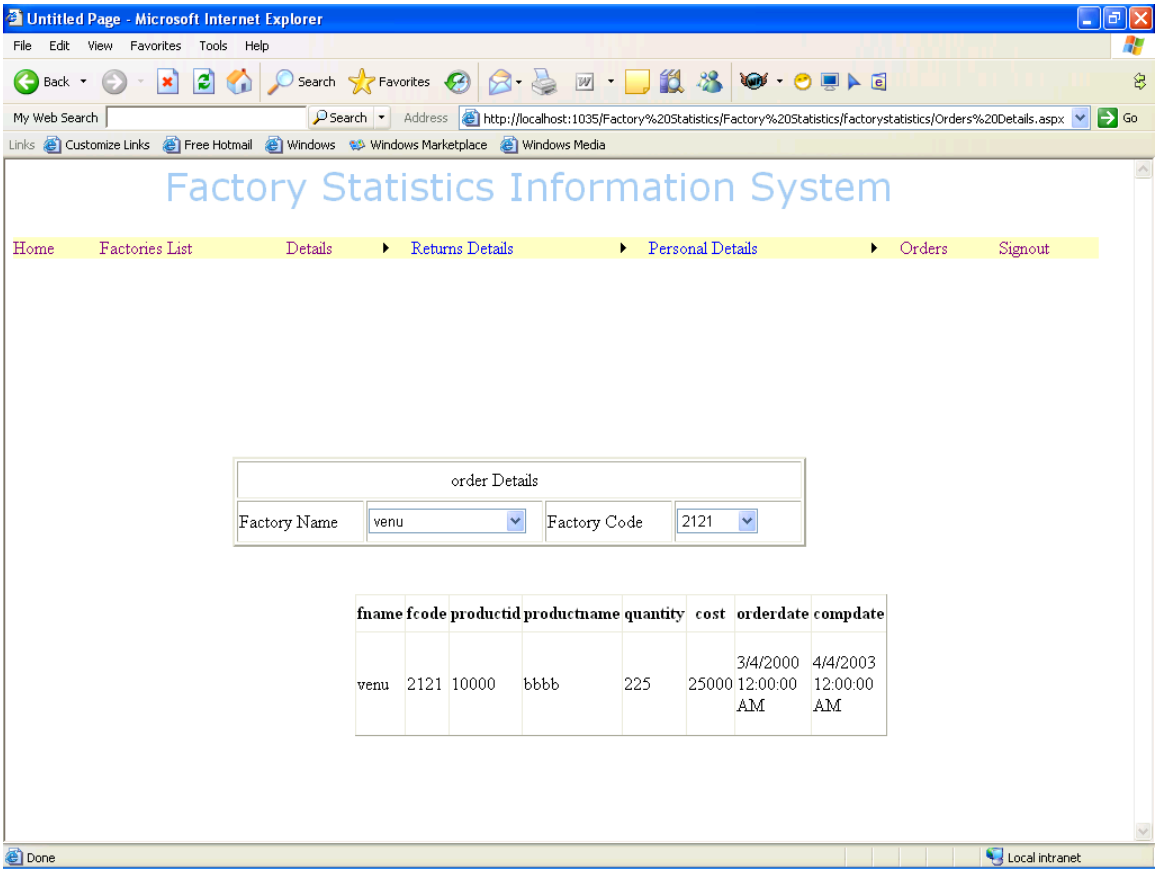
COMMITTEE DETAILS



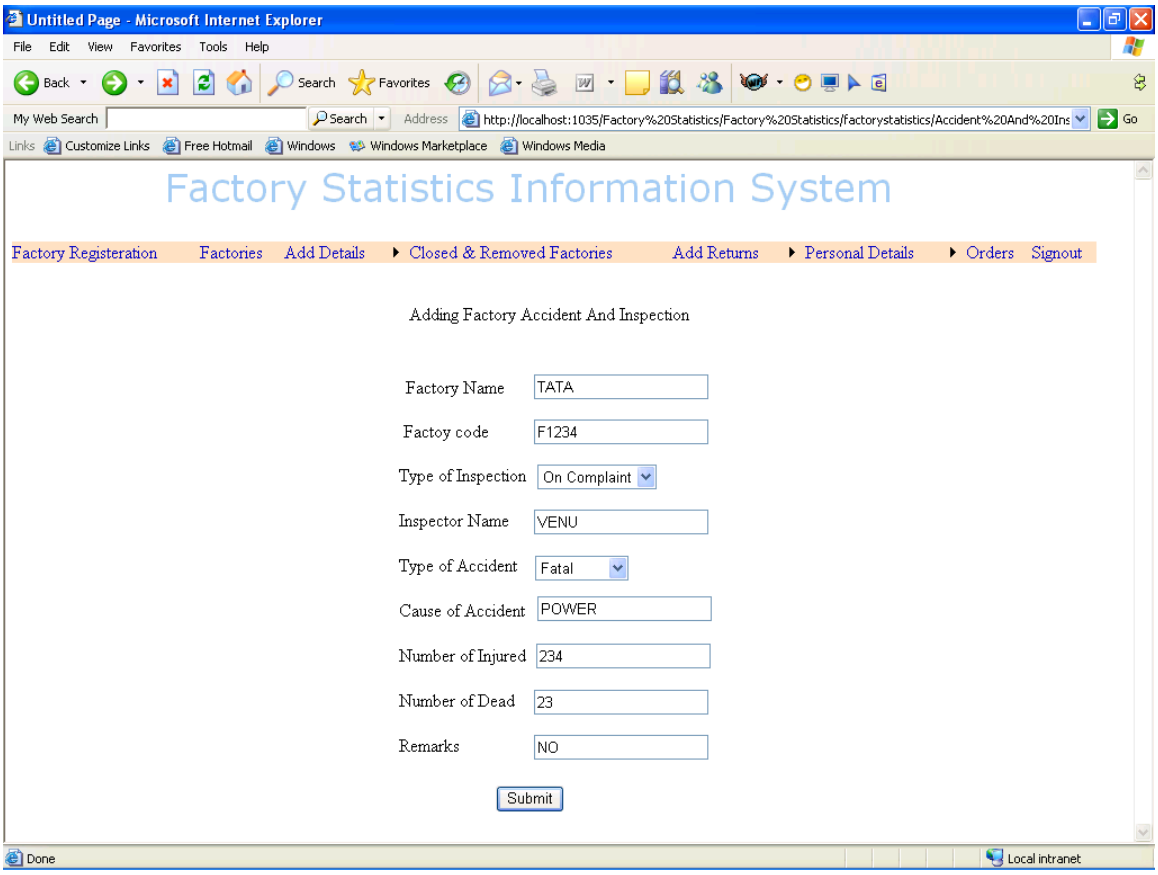
PARTNERS DETAILS



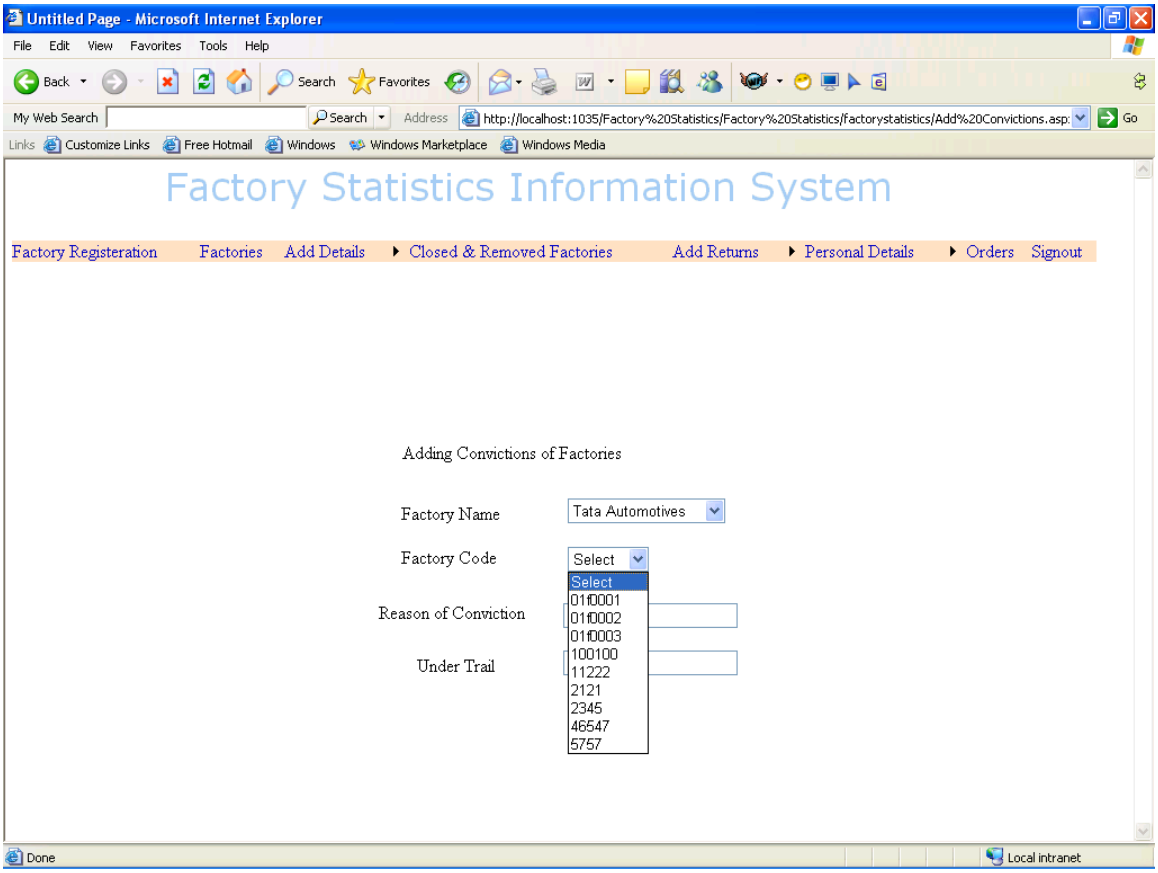
EMPLOYEE DETAILS



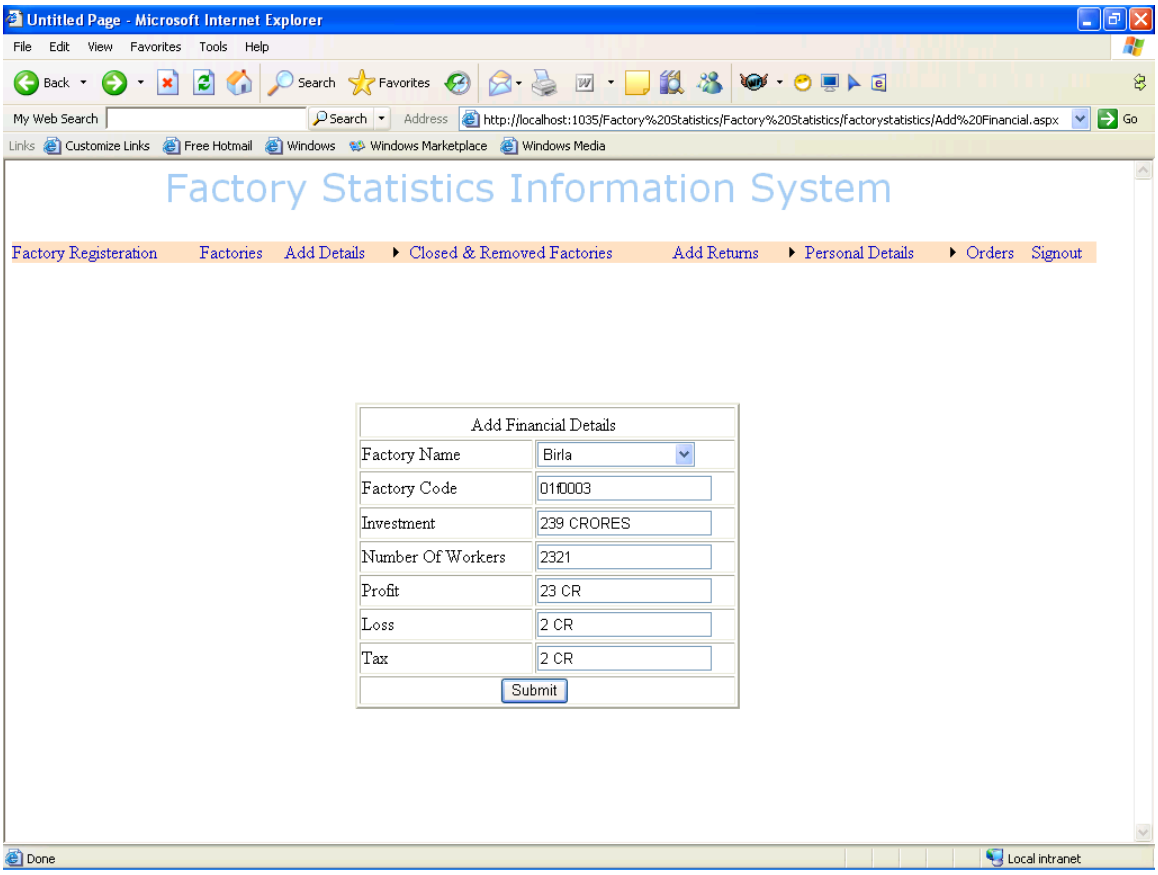
ORDERS DETAILS



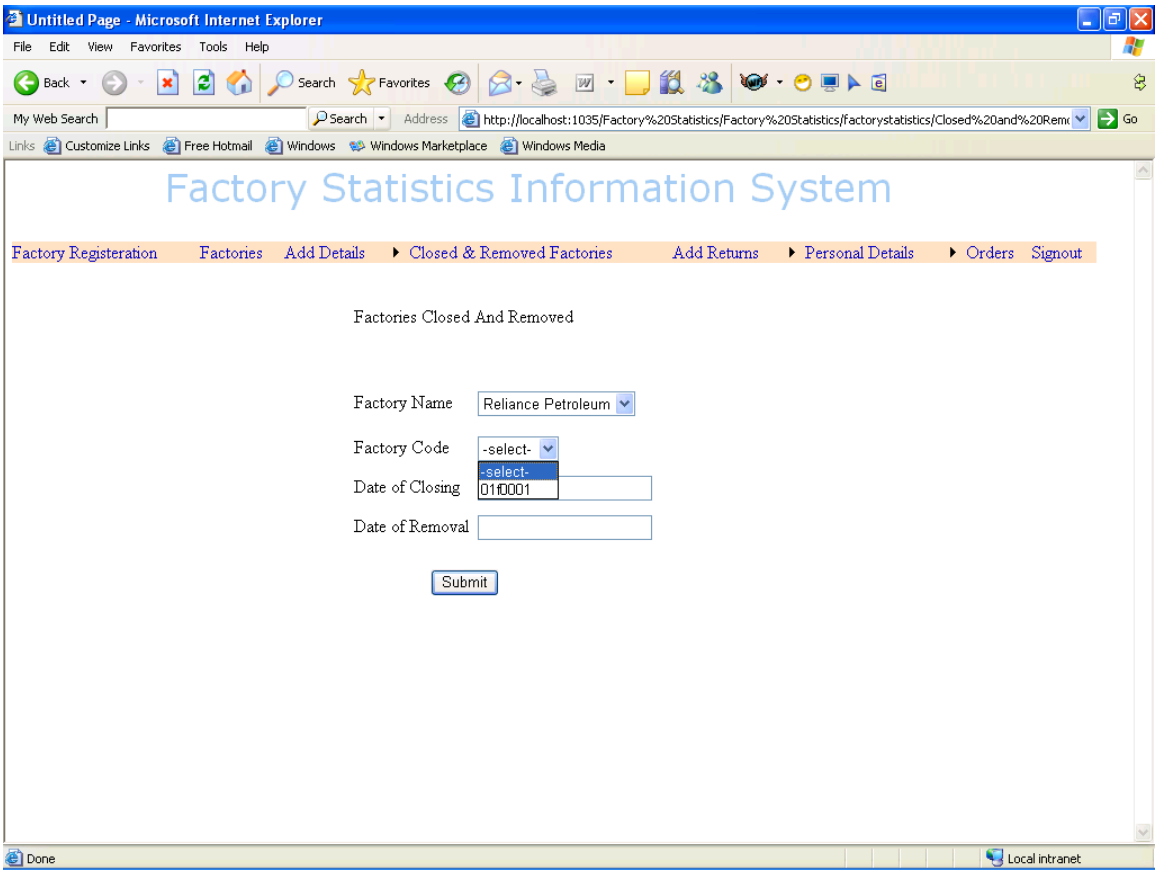
ADDING FACTORY ACCIDENT & INSPECTION



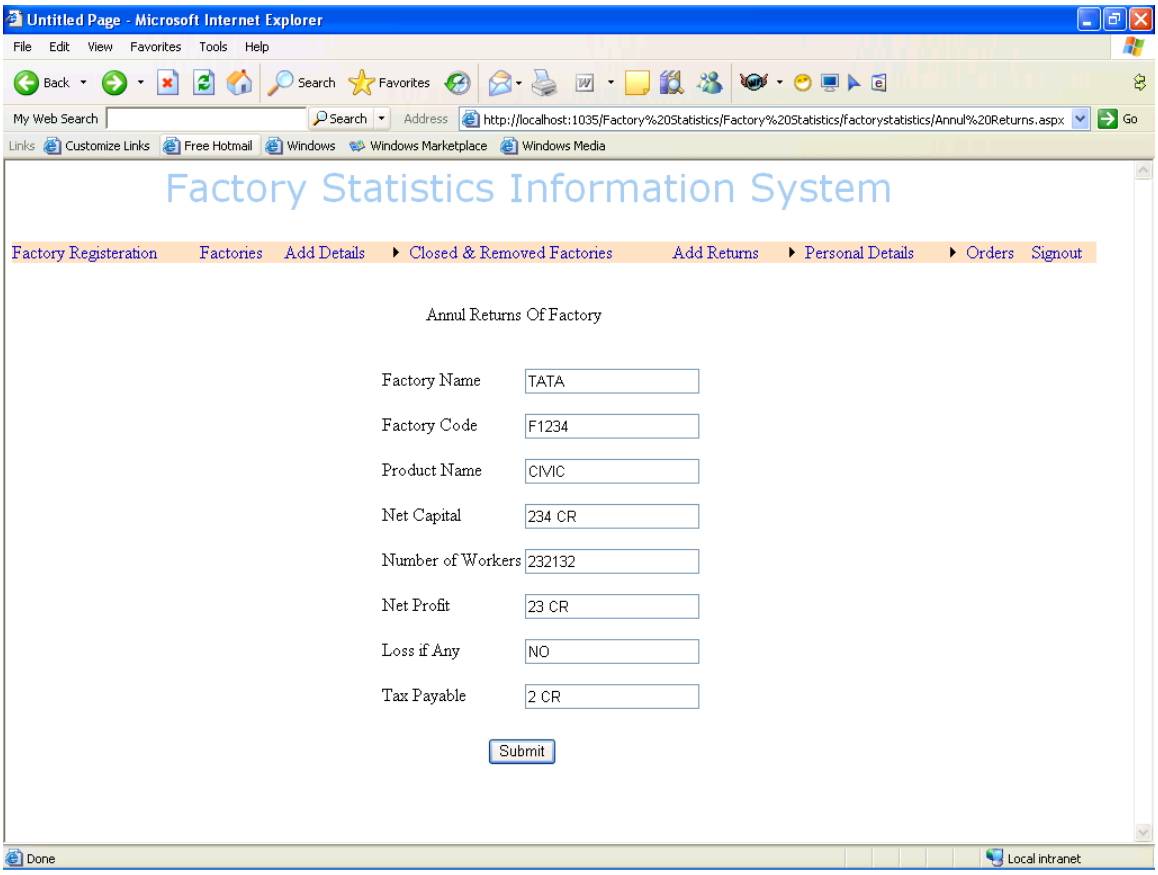
ADDING CONVICTIONS OF FACTORY



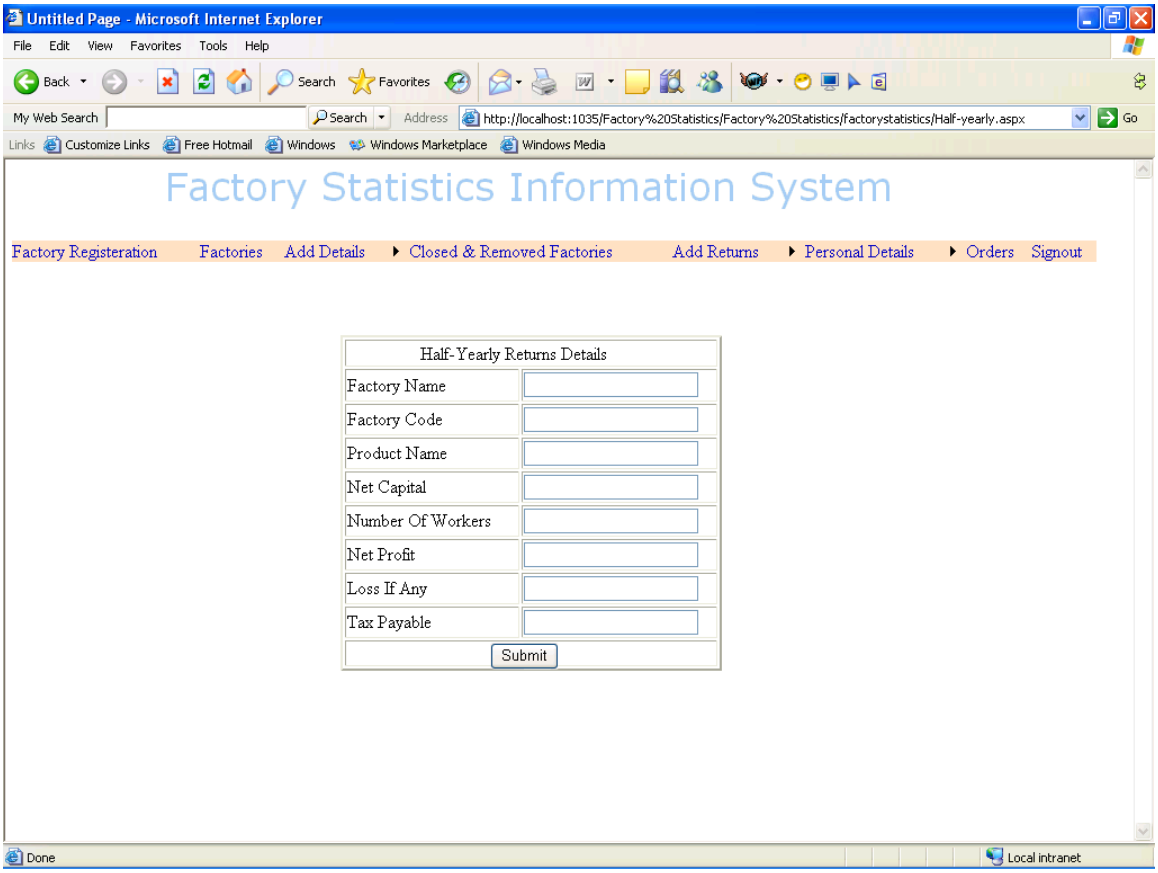
ADDING FINANCIAL DETAILS



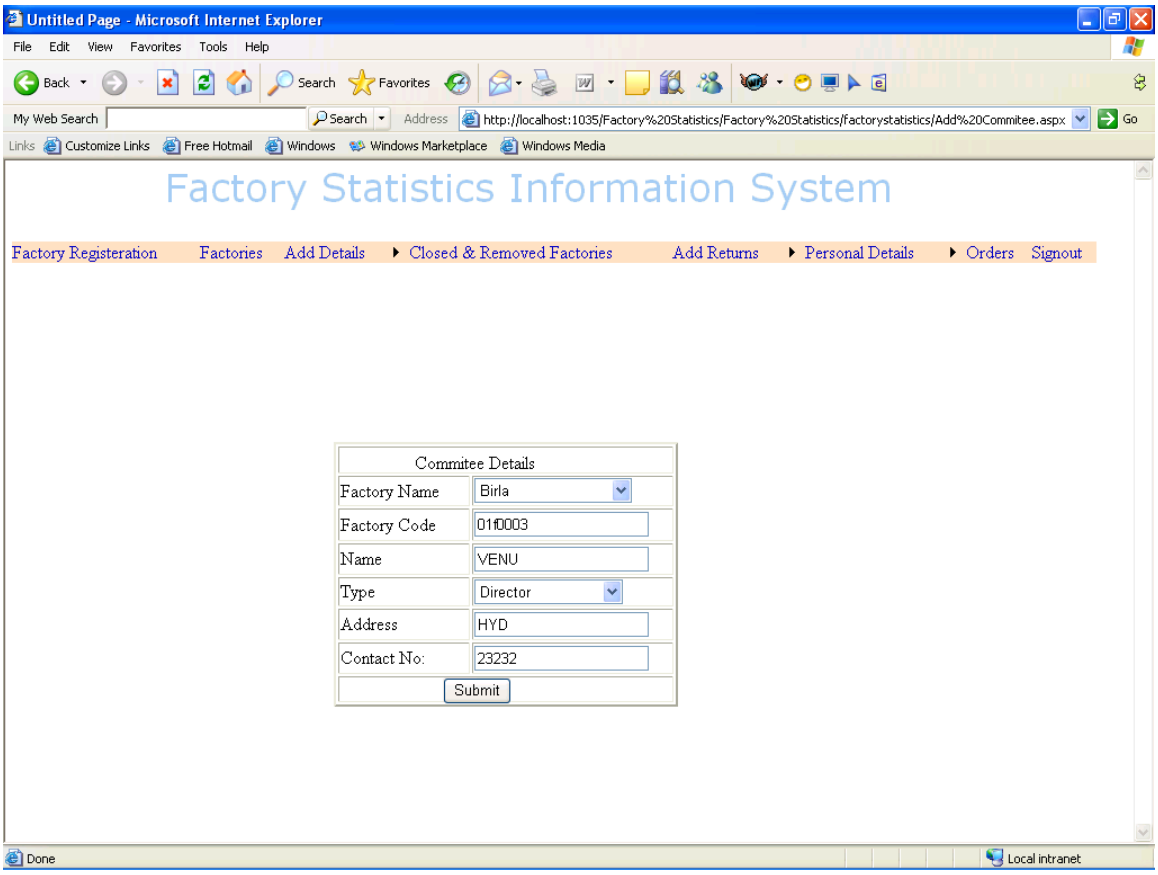
ADDING CLOSED & REMOVED FACTORIES



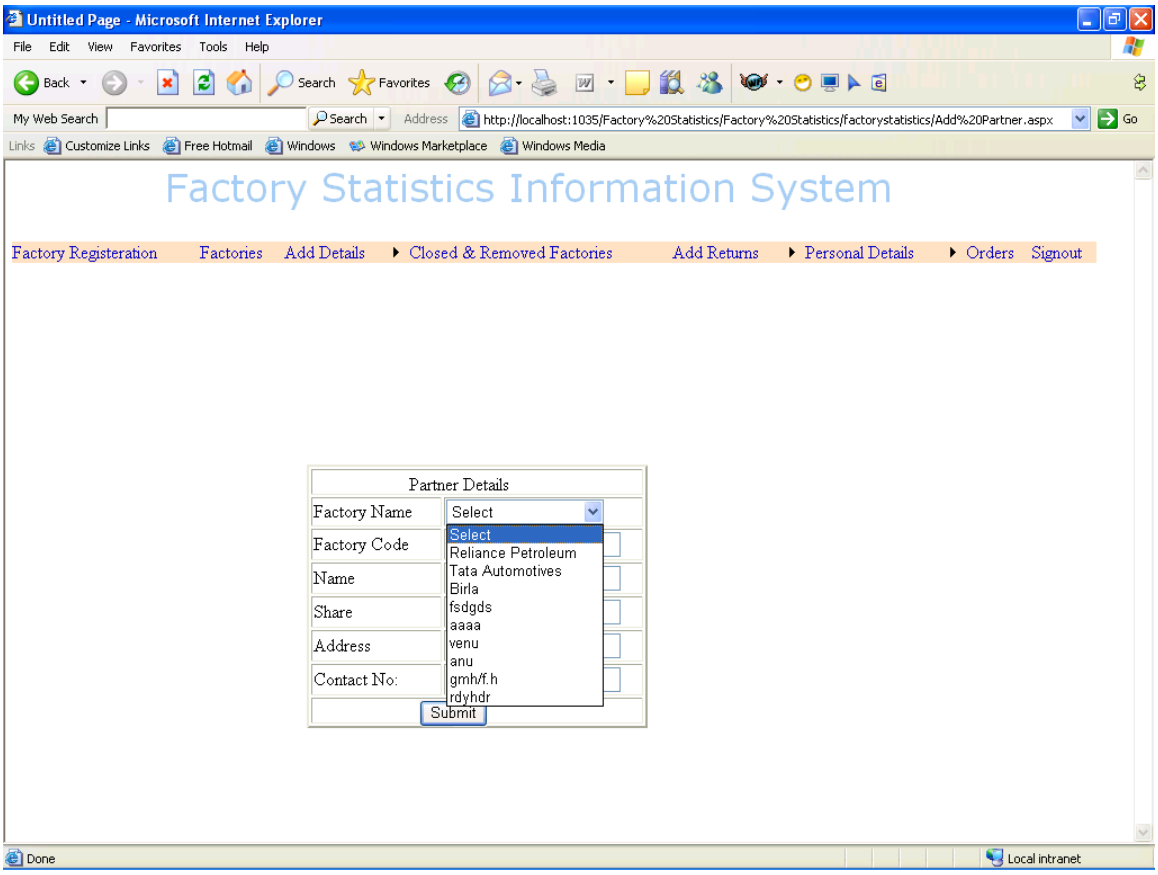
ADDING ANNUAL RETURNS



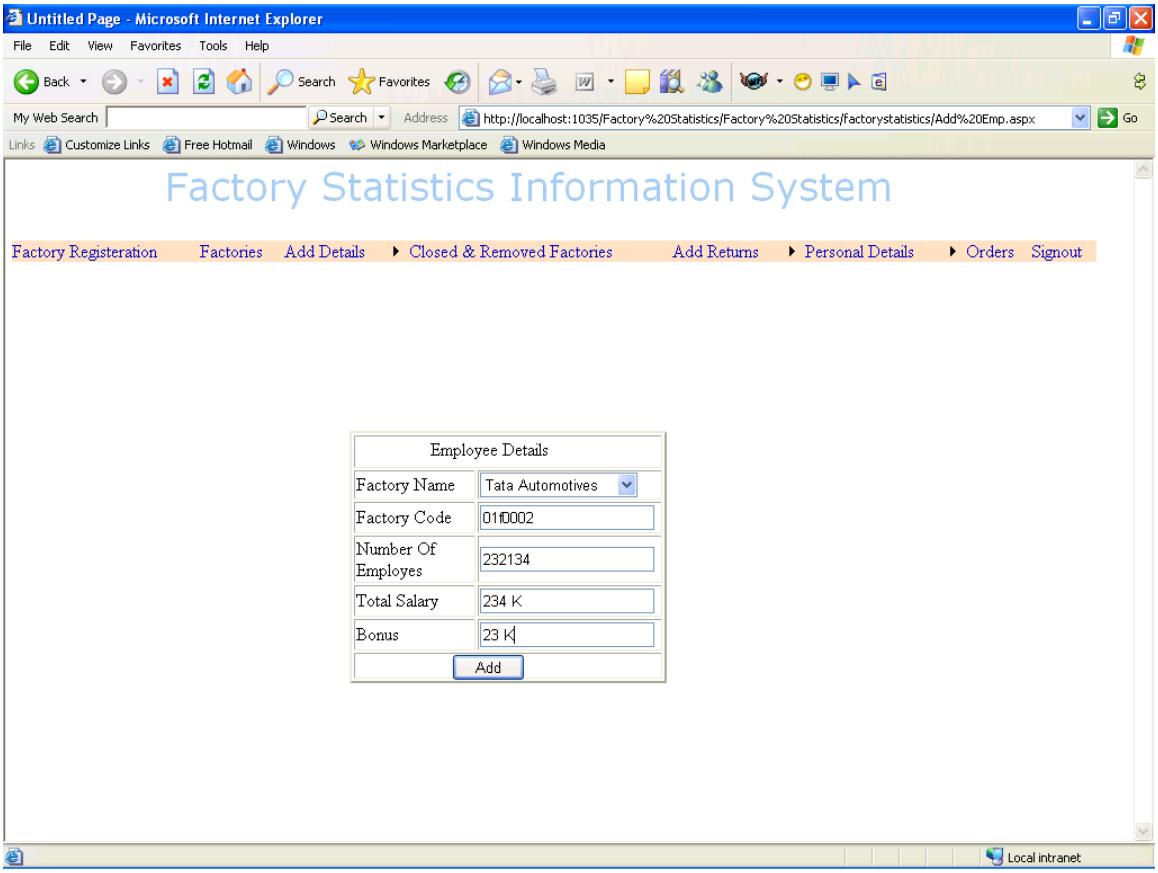
ADDING HALF YEARLY RETURNS



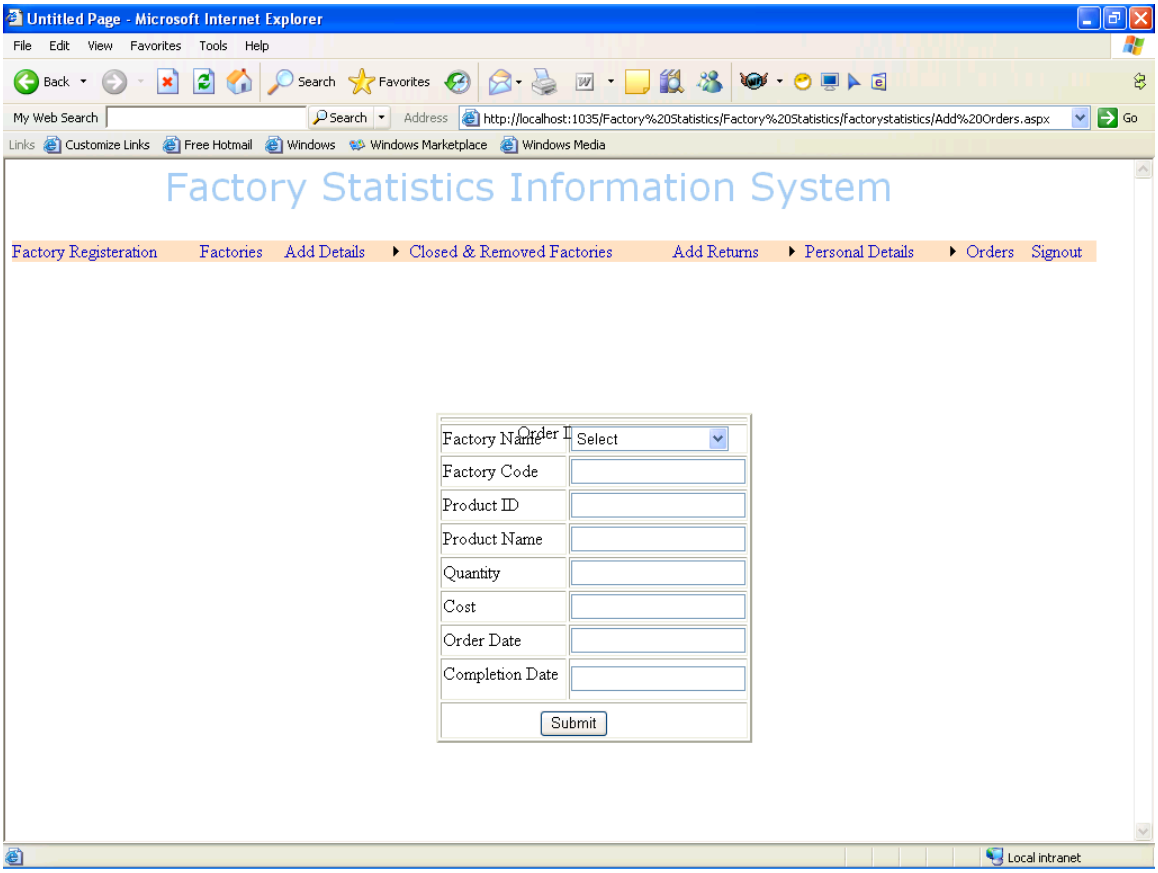
ADDING COMMITTEE DETAILS



ADDING PARTNERS DETAILS



ADDING EMPLOYEE DETAILS



OREDERS FOR A FACTORY

Future Enhancements

CONCLUSION:

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.

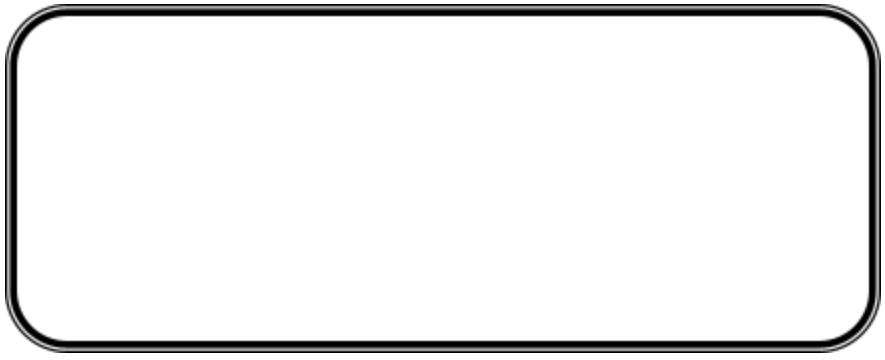

Future Enhancements

FUTURE ENHANCEMENTS:

This application avoids the manual work and the problems concern with it. It is an easy way to obtain the information regarding the various products information that are present in the Super markets.

Well I and my team members have worked hard in order to present an improved website better than the existing one's regarding the information about the various activities. Still ,we found out that the project can be done in a better way. Primarily, when we request information about a particular product it just shows the company, product id, product name and no. of quantities available. So, after getting the information we can get access to the product company website just by a click on the product name .

The next enhancement that we can add the searching option. We can directly search to the particular product company from this site .These are the two enhancements that we could think of at present.



BIBLIOGRAPHY

The following books were referred during the analysis and execution phase of the project

MICROSOFT .NET WITH C#
Microsoft .net series

ASP .NET 2.0 PROFESSIONAL
Wrox Publishers

ASP .NET WITH C# 2005
Apress Publications

C# COOK BOOK
O reilly Publications

PROGRAMMING MICROSOFT ASP .NET 2.0 APPLICATION
Wrox Professional Guide

BEGINNING ASP .NET 2.0 E-COMMERCE IN C# 2005
Novice to Professional.

WEBSITES:

www.google.com
www.microsoft.com