



NEAR EAST UNIVERSITY

FACULTY OF ENGINEERING

DEPARTMENT OF COMPUTER ENGINEERING

**CARGO & TRANSPORTATION APPLICATION
SOFTWARE PROGRAM**

**Graduation Project
COM- 400**

**Students : Göktuğ Ataç (20020450)
 Sinan Özerenler (20010749)**

Supervisor : Ümit Soyer

Nicosia – 2006



ACKNOWLEDGEMENT

We, Göktuğ Ataç and Sinan Özerenler are dedicated this thesis to our families who have supported us at rough times and have shared our joy at good times through all these months.

First, we owe the greatest thanks to our supervisor, Mr. Ümit Soyer, whose encouragement and enthusiasms have been constant sources of inspiration to us. Also his invaluable advices and belief in our work and ourself over the course of this Graduation Project. He is one of the best advisor a student can wish, both as a person and as a mentor.

Second, we would like to express our gratitude to Near East University (NEU) for the scholarship that made the work possible. Also special thanks starting from our Dean Prof. Dr Fakhreddin Mamedov , our department chairman Prof. Dr. Doğan İbrahim , Assoc. Prof. Dr Rahib Abiyev and all other lecturers for the advices and helps.

Finally, we would like to thank all our friends for their advices , supports and being with us with a big respect and patience.

ABSTRACT

Designing software programs is a computer & software Engineering responsibility. Increasing the complexity of the technological process the importance of software and engineers is increasing in the parallel way of technology too. Engineering is the application of science to the needs of humanity. This is accomplished through the application of knowledge, mathematics, and practical experience to the design of useful objects or processes. Professional practitioners of engineering are called engineers.

The aim of our project is to develop an office software program as in the way of the definition of engineering for a cargo & transportation company. For this purpose the structure, functions and analysis of its main blocks of cargo & transportation company is described.

The main problem of this software program is making analysis and constructing the knowledge base detailed. For example if we look from the point of a customer what are the main necessities of a cargo and transportation company. This program is designed in the way of necessities of a company. As we said company has two main department cargo and transportation. Briefly cargo department is responsible with receiving and sending cargo. The transportation department is responsible with transporting people in desired destinations.

Finally, in this technologically fastly growing century we try to create a new approach which we expect to bring speed, reliability, flexibility to or with systems and users in the company.

TABLE OF CONTENTS

ACKNOWLEDGEMENT	i
ABSTRACT	ii
TABLE OF CONTENTS	iii
LIST OF ABBREVIATIONS	iv
INTRODUCTION	1
CHAPTER ONE: AN OVERVIEW OF MICROSOFT VISUAL STUDIO .NET & SQL SERVER	2
1.1 History of Visual Studio	2
1.1.1 Visual Studio 97	2
1.1.2 Visual Studio 6.0	2
1.1.3 Visual Studio .NET (2002)	3
1.1.4 Visual Studio .NET 2003	4
1.1.5 Visual Studio 2005	4
1.1.6 Future versions	5
1.2 Features of Visual Studio 2003	6
1.2.1 New Features	6
1.2.2 Power Toys	8
1.2.3 System Requirements	8
1.3 Product Overview for Visual Studio .NET 2003	10
1.3.1 Enterprise Architect	10
1.3.2 Enterprise Developer	12
1.3.3 Professional	13
1.3.4 Academic	14
1.4 An Overview to C# Programming Language	16
1.4.1 Architectural History	16
1.4.2 Object Oriented Programming	17
1.4.3 User Defined Classes in C#	19
1.4.4 Language features	19
1.4.5 C# 2.0 & 3.0 New Language Features	21
1.5 Overview of SQL Server Developer Edition	24

1.5.1 Understanding the Definition of Database	24
1.5.2 Database Architecture And Features of SQL 2000	26
1.5.3 Ms-SQL Server Permissions and Uses	29
CHAPTER TWO : PROJECT ANALYSIS & OVERVIEW	
2.1 Project Overview	34
CHAPTER THREE: EXAMINATION OF CODE STRUCUTRE	
3.1 Selecting Data From Database	56
3.2 Inserting Data to Database	56
3.2.1 Using Stored Procedures	56
3.2.2 Using SQL Query	57
3.3 Updating Data in Database	58
3.4 Delete Data From Database	58
3.5 Windows Control	59
3.5.1 Adding Item to ComboBox	59
3.5.2 Filling DataGrid	59
3.5.3 Filling ListBox	60
3.5.4 Creating DialogBox	61
CHAPTER FOUR: WEB APPLICATION PART OF PROJECT	
4.1 HTML and XHTML	62
4.2 OVERVIEW OF ASP.NET	64
4.3 Web Project Overview	66
CONCLUSION	70
REFERENCES	71

LIST OF ABBREVIATIONS

NEU	Near East University
C#	C Sharp
DBMS	Data Base Management System
SQL	Structured Query Language
ASP	Active Server Pages
DIME	Direct Internet Messaging
EIF	Enterprise Implementation Framework
LINQ	Language Integrated Query
DRI.	Declarative Referential Integrity

INTRODUCTION

Software or Computer engineering is the profession of people who create and maintain software applications by applying technologies and practices from computer science, project management, engineering, application domains and other fields. They deal with matters of cost and reliability, like traditional engineering disciplines. In spite of the enormous economic growth and productivity gains enabled by software, persistent complaints about the quality of software remain. So the question of "What is the best way to make more and better software?" is still a problem.

In our project we wanted to create a new approach for this question with our analysis and structures. First of all we used a new and fastly growing Microsoft .NET platform in our project. We analysed the companies of structures in detailed. These are the main objectives of our project.

Our projects consist of introduction ,four chapters and conclusion. We seperated project in the main topics of chapters as our programming language, database language, about structure and the main parts , main parts details, website part of project.

Chapter one describes the programming language platform Microsoft Visual Studio .Net 2003 Enterprise Developer platform and our programming language C#(C Sharp). Also it presents the database Microsoft SQL Server 2000 Developer Edition that is used in our project.

Chapter two presents construction period , structure and the main parts of cargo and transportation departments in summary. We describe the architecture of program, the steps , and the diffucilties that we encountered.

Chapter three is devoted to parts detailed that we described in chapter two with main topics. Actually, what type of codes is needed and how we construct the project harmony with the platform and with database are described. Also the compatiblity of program with differnet operating systems are mentioned.

Chapter four presents results of program as summary and starts to describe the website of the company ,a briefly about HTML and ASP.NET platforms, its construction period, its main work and the duty that supplied over the internet.

CHAPTER 1

AN OVERVIEW OF MICROSOFT VISUAL STUDIO .NET

1.1 History of Visual Studio :

1.1.1 Visual Studio 97

Microsoft first released Visual Studio in 1997, bundling together many of its programming tools for the first time. Visual Studio 97 was released in two editions, Professional and Enterprise. It included Visual Basic 5.0 and Visual C++ 5.0, primarily for Windows programming; Visual J++ 1.1 for Java and Windows programming; and Visual FoxPro 5.0 for xBase programming. It introduced Visual InterDev for creating dynamically generated web sites using Active Server Pages. A snapshot of the Microsoft Developer Network library was also included. Visual Studio 97 was Microsoft's first attempt at using the same development environment for multiple languages. Visual C++, J++, InterDev, and the MSDN Library all used one environment, called Developer Studio. Visual Basic used a separate environment, as did Visual FoxPro.

1.1.2 Visual Studio 6.0

The next version, version 6.0, was released in 1998. The version numbers of all of its constituent parts also moved to 6.0, including Visual J++ which jumped from 1.1, and Visual InterDev which was at 1.0. This version was the basis of Microsoft's development system for the next four years, as Microsoft transitioned their development focus to the .NET Framework.

Visual Studio 6.0 was the last version to include Visual Basic as most VB programmers knew it; subsequent versions would include a quite different version of VB based on .NET. It was also the last version to include Visual J++, which included deeper ties to Windows and proprietary extensions to the Java language that were incompatible with Sun's version. This caused Sun to sue Microsoft. As part of the settlement, Microsoft would no longer sell programming tools that targeted the Java Virtual Machine.

Although Microsoft's long-term goal was to unify its tools under one environment, this version actually had one more environment than VS 97. Visual J++ and Visual InterDev broke away from the Visual C++ environment, while Visual Basic and Visual FoxPro maintained their separate tools.

1.1.3 Visual Studio .NET (2002)

Microsoft released Visual Studio .NET in 2002 (the beta version was released on the Microsoft developer network in 2001). The biggest change was the introduction of a managed code development environment using the .NET Framework. Programs developed using .NET are not compiled to machine language (like C++ is, for example) but instead to a format called MIL or CIL. When a MIL application is executed, it is compiled while being executed into the appropriate machine language for the platform it is being executed on, thereby making code portable across multiple platforms. Programs compiled into MIL can be executed only on platforms which have an implementation of Common Language Infrastructure. It is possible to run MIL programs in Linux or Mac OS X using non-Microsoft .NET implementations like Mono and DotGNU.

Microsoft introduced C#, a language similar to Java, that targets .NET. It also introduced the successor to Visual J++ called Visual J#. Visual J# programs use Java's language syntax. However, unlike Visual J++ programs, Visual J# programs can only target the .NET Framework, not the Java Virtual Machine that all other Java tools target.

Visual Basic was drastically changed to fit the new framework, and the new version was called Visual Basic .NET. Microsoft also added extensions to C++, called Managed Extensions for C++, so that C++ programmers could create .NET programs.

Visual Studio .NET can be used to make applications targeting Windows (using Windows Forms, part of the .NET Framework), Web (using ASP.NET and Web Services) and, with an add-in, portable devices (using the .NET Compact Framework) like compatible mobiles and laptops.

The Visual Studio .NET environment was rewritten to partially use .NET. All languages are unified under one environment, except for Visual FoxPro. Compared to previous versions of Visual Studio, it has a cleaner interface and greater cohesiveness. It is also more customizable with tool windows that automatically hide when not in use.

Also in this version, Visual FoxPro was no longer being bundled and is now sold separately. The

internal version number of Visual Studio .NET is version 7.0.



1.1.4 Visual Studio .NET 2003

Microsoft introduced a minor upgrade to Visual Studio .NET in 2003 called Visual Studio .NET 2003. At that point, it referred to the previous version as Visual Studio .NET 2002. It included an upgrade to the .NET Framework, version 1.1. It also came with built-in support for developing programs for mobile devices, using either ASP.NET or the .NET Compact Framework. As well, the Visual C++ compiler was improved to be more standards-compliant, especially in the area of partial template specialization, and a free version of the same C++ compiler shipped with Visual Studio .NET 2003 was made available to the public, although without the IDE, called Visual C++ Toolkit 2003 (no longer available and now superseded by the Express Editions).

Visual Studio 2003 shipped in four editions: Academic, Professional, Enterprise Developer, and Enterprise Architect. The Enterprise Architect edition included an implementation of Microsoft Visio's modeling technologies, which focused on creating Unified Modeling Language-based visual representations of an application's architecture. "Enterprise Templates" were also introduced, to help larger development teams standardize coding styles and enforce policies around component usage and property settings.

The internal version number of Visual Studio .NET 2003 is version 7.1.

1.1.5 Visual Studio 2005

Visual Studio 2005, codenamed Whidbey (a reference to NAS Whidbey Island in Puget Sound), was released online in October 2005 and hit the stores a couple of weeks later. Microsoft removed the ".NET" moniker from Visual Studio 2005 (as well as every other product with .NET in its name), but it still primarily targets the .NET Framework, which was upgraded to version 2.0.

The most important C# language feature added in this version was the introduction of generics, which are very similar to C++ templates. This potentially increases the number of bugs caught at compile-time instead of run-time. C++ also got a similar upgrade with the addition of C++/CLI which is slated to eventually replace Managed C++.

Other new features of Visual Studio 2005 include the "Deployment Designer" which allows application designs to be validated before deployments, an improved environment for web publishing when combined with ASP.NET 2.0 and load testing to see application performance under various sorts of user loads.

Visual Studio 2005 also added extensive 64-bit support. Visual C++ 2005 supports compiling for x64 (AMD64 and EM64T) as well as IA-64 (Itanium). Previous versions of Visual Studio did not come with 64-bit support. The Platform SDK only included the 64-bit compilers and 64-bit versions of the Visual C++ 6.0 libraries. The 64-bit versions of the Visual C++ .NET 2003 (7.1) libraries were available only by e-mailing Microsoft with this email address: libs7164@microsoft.com.

Visual Studio 2005 is available in several editions, which are significantly different than previous versions: Express, Standard, Professional, Tools for Office, and Team System. In addition to these, four separate Team System editions are provided in conjunction with MSDN Premium subscriptions: Team Suite, Team Edition for Software Architects, Team Edition for Software Developers, and Team Edition for Software Testers.

Team System includes support for large development organizations, and comes in separate editions for software architects, developers, and testers.

Tools for the Microsoft Office System lets developers create extensions to Microsoft Office.

Express Editions were introduced for amateurs, hobbyists, and small businesses, and are available as a free download from Microsoft's web site. There are Express Editions for each language (Visual Basic, Visual C++, Visual C#, Visual J#), each targeting the .NET Framework on Windows, as well as a Visual Web Developer for creating ASP.NET web sites. The Express Editions lack many of the more advanced development tools and extensibility of the other editions. Individual language editions of Visual Studio are no longer available.

Compared to the previous versions of Visual Studio, this time its interface has come to closely resemble the IBM Eclipse interface (particularly IBM IDEs' distinctive notched tab corners).

Visual Studio 2005's internal version number is 8.0.

1.1.6 Future versions

The successor to Visual Studio 2005, code-named *Orcas*, is currently under development. The name *Orcas* is, just like Whidbey, a reference to an island in Puget Sound, namely Orcas Island. The successor to *Orcas* is code-named *Hawaii*. Visual Studio has a facility for developers to write extensions (or add-ins) for Visual Studio to extend its capabilities. These add-ins "plug into" Visual Studio and offer some benefits not available from Visual Studio itself.

1.2 Features of Visual Studio 2003

1.2.1 New Features

VS 2003 includes plenty of new features. The following sections describe what we think are the best ones.

Java Support: VS 2003 brings first-time integration with Visual J#, a tool for Java developers who want to use Microsoft's .NET framework to build applications and XML Web services. Outside of Visual J#, VS 2003 also supports Visual Basic (VB), Visual C++, and Visual C# developers. The new IDE is available in Standard, Professional, Enterprise Developer, and Enterprise Architect editions.

.NET 1.1 Support: If you want to build applications supporting Microsoft's new .NET Framework 1.1, VS 2003 appears to be an absolute must. According to Microsoft, .NET 1.1 provides increased scalability with support for up to 32 processors, along with improvements such as better mobile and database support. It also supports IPv6 and, for the ASP.NET platform only, tighter permissions lockdown through code access security.

Side-by-Side Development: For easier migration to the new development environment, you can run VS 2003 and .NET 1.1 side by side with VS 2002 and .NET 1.0. This new capability is enabled by side-by-side execution, an enhancement in version 1.1 that permits execution of multiple versions of applications or components on the same computer.

However, successful side-by-side development assumes you have adequate disk space to run both frameworks and both IDEs. VS 2003 alone requires 900MB of available disk space on the system drive and 3.3GB of available space on the installation drive.

VS 2003 also includes a setting that lets you compile for either .NET 1.1 or .NET 1.0. Be careful, however, not to compile code to version 1.0 that requires brand-new capabilities in version 1.1.

You should also be aware that VS 2003 uses different file extensions than VS 2002. If you open a project already started in VS 2002, a pop-up appears informing you that the 2002 project is about to be converted to VS 2003 format.

Native Mobile Support: For VB and C# developers, VS 2003 comes with two types of native mobile support. By using VS 2003's built-in Compact Framework—together with Microsoft Windows Forms Designer—you can create applications for mobile devices

running on the Pocket PC, the Pocket PC Phone Edition, or Windows CE .NET operating system. Alternatively, VB and C# developers can use VS 2003's integrated ASP.NET Web Forms—along with VS .NET Designer—to build thin-client applications for more than 200 Web site META-enabled devices. The supported devices include WAP phones, pagers, and PDAs.

Better Database Support: VS 2003 also come with new managed data provider tools, for easier connectivity to Microsoft SQL Server, Microsoft Jet, Oracle, and ODBC databases.

Quicker Development: Also in VS 2003, Microsoft introduces enhancements to both code completion and IntelliSense. IntelliSense, a design-time assistant, now appears in the immediate window, for easier access.

Here's another time-saver that Microsoft has been demonstrating to developers and journalists. If you type Try at the beginning and end of the handling block, VS automatically inserts the stub code for the rest of the block.

Web Services: Other new tools in VS 2003 support the latest Web services standards, including WS-Security, WS-Routing, WS-Attachments, and Direct Internet Messaging (DIME).

Code Obfuscation: Also new in VS 2003 is a built-in utility for source code obfuscation. In addition, VS developers can now download Microsoft's Enterprise Implementation Framework (EIF), a tool that the company promises lets you quickly add runtime monitoring to your applications. The source code obfuscation tool, PreEmptive Dotfuscator Community Edition, is the "lite" version of a product from PreEmptive Solutions, a Microsoft ISV partner based in Cleveland. Microsoft says it included the tool to protect the intellectual property of programmers who distribute VS 2003 code, as well as to help developers reduce the size and improve the performance of mobile and other .NET applications. The tool is designed to render Microsoft Intermediate Language (MSIL), which is said to be very difficult, if not impossible, to reverse-engineer into comprehensible source code.

Improved Upgrade Wizard: Microsoft improved the VS Upgrade Wizard, a tool for VB developers only, with first-time support for user controls and Web classes. The enhanced Upgrade Wizard is designed to automatically upgrade Visual Basic 6.0 projects when opened in VS 2003. The tool modifies the language for syntax changes, while also converting VB 6.0 forms. The wizard also notifies developers when manual

changes need to be made to code. To view the code statement, you navigate to the Task List window and double-click on the task item.

1.2.2 Power Toys

Microsoft's Power Toys work with VS 2003 only. The latest release in the series is VSMousebindings, a tool for assigning mouse buttons to Visual Studio commands. Earlier Power Toy releases for VS 2003 are as follows:

- * VSWindowsManager, for designing custom window layouts
- * VB Commenter, for quickly adding comments to VB source code
- * VSTweak, for modifying some of the most obscure VS options and settings
- * VSEdit, a command-line tool that lets you load files into a currently running instance of VS 2003
- * Custom Help Builder, for producing XML-based Visual Basic or Visual C# help files that can be fully integrated into VS 2003

Some VS 2003 users have complained online to Microsoft that, after installing Custom Help Builder, they get notifications that the tool is not installed. Microsoft has promised a Custom Help Builder fix.

Meanwhile, Microsoft is readying two more Power Toys for VS 2003:

- * VSCMShell Window, a tool that will permit use of the same window for accessing both the VS Commands window and the external CMD.exe process
- * VSTransparency, enabling creation of "semi-transparent" floating windows and pop-ups so that developers can see through to the code beneath.

1.2.3 System Requirements

To use Visual Studio .NET 2003, you need :

<i>Processor</i>	450-megahertz (MHz) PentiumII-class processor, 600-MHz Pentium III-class processor recommended
<i>Operating System</i>	Visual Studio .NET 2003 can be installed onto any of the following systems:

- Microsoft Windows® Server 2003
- Windows XP Professional
- Windows XP Home Edition¹
- Windows XP Media Center Edition
- Windows XP Tablet PC Edition
- Windows 2000 Professional (SP3 or later **required** for installation)
- Windows 2000 Server (SP3 or later **required** for installation)

Applications can be deployed onto the following systems:

- Windows Server 2003
- Windows XP Professional
- Windows XP Home Edition
- Windows XP Media Center Edition
- Windows XP Tablet PC Edition
- Windows 2000 (Service Pack 2 recommended)
- Windows Millennium Edition (Windows Me)
- Windows 98
- Microsoft Windows NT® 4.0 (Service Pack 6a required)
- Windows 95 (using Microsoft Visual C++® .NET)

Memory

- Windows Server 2003: 160 megabytes (MB) of RAM
- Windows XP Professional: 160 MB of RAM
- Windows XP Home Edition: 96 MB of RAM
- Windows XP Media Center Edition: 160 MB of RAM
- Windows XP Tablet PC Edition: 160 MB of RAM
- Windows 2000 Professional: 96 MB of RAM
- Windows 2000 Server: 192 MB of RAM

Hard Disk

- 900 MB of available space required on system drive, 3.3 gigabytes (GB) of available space required on installation drive
- Additional 1.9 GB of available space required for optional MSDN Library documentation

Drive CD-ROM or DVD-ROM drive

Display Super VGA (1024 x 768) or higher-resolution display with 256 colors

1.3 Product Overview for Visual Studio .NET 2003

Introducing Visual Studio .NET 2003—visionary yet practical, the single comprehensive development tool for creating the next generation of applications has arrived. Developers can use Visual Studio .NET to:

- Build the next-generation Internet.
- Create powerful applications quickly and effectively.
- Span any platform or device.

Visual Studio .NET is the only development environment built from the ground up to enable integration through XML Web services. By allowing applications to share data over the Internet, XML Web services enable developers to assemble applications from new and existing code, regardless of platform, programming language, or object model. Visual Studio .NET 2003 is available in the following editions:

1.3.1 Enterprise Architect

Visual Studio .NET 2003 provides developers with comprehensive tools for designing and building distributed applications for Microsoft Windows®, the Web, and mobile devices. Visual Studio .NET 2003 Enterprise Architect (VSEA) builds on the power of Visual Studio .NET 2003 Enterprise Developer by including additional capabilities for designing, specifying, and communicating application architecture, development best practices, and application functionality. Developers using Visual Studio .NET 2003 Enterprise Architect will benefit from the ability to:

- * *Visually model applications, databases, and business processes.* Clearly define application functionality and architecture for XML Web services and applications, and visually orchestrate business processes.
- * *Create sound architectural frameworks and best practices guidelines.* Increase application development efficiency through starting application frameworks, sharing best practices guidelines, and simplifying development and management of complex applications.
- * *Build on a scalable and dependable platform for distributed applications.* Create secure, reliable, and high performance applications using the Visual Studio .NET 2003 integrated development environment (IDE) and the Windows Server 2003 programming model. Choose from a broad range of integrated third-party tools.

Visually Model Applications, Databases, and Business Processes use a complete set of Microsoft Visio®—based modeling capabilities to create and communicate application architecture, business requirements, database design, and business processes. Architects using Microsoft Visual C++® .NET, Microsoft Visual Basic® .NET, or Microsoft Visual C#® .NET can use Unified Modeling Language (UML) models to specify application architecture and functionality, reduce development time by directly generating classes, functions, and methods, and document existing application code by reverse-engineering projects. Visual Studio .NET 2003 Enterprise Architect enables developers to create architectural designs and models that they can share with the rest of their team.

Visual Studio .NET 2003 Enterprise Architect provides end-to-end support for database modeling, including conceptual, logical, and physical views. Business analysts can easily enter business rules using the Fact Editor, which in turn generates an underlying database model that can be refined by a database analyst. Roundtrip engineering guarantees that changes made at any of the views will be reflected throughout, improving communication across the development team. Business processes can be defined and orchestrated using a full-featured version of Microsoft BizTalk® server designed for developers. This makes it easier to compose applications from existing functionality—whether that functionality is internal or external to an organization.

Create Sound Architectural Frameworks and Best Practices Guidelines Enterprise Templates and the Template Description Language can help developers meet the challenges of rapidly proliferating innovations and technologies, and increase collaboration across development teams. Architects can use Enterprise Template projects to create application starting points by specifying an initial application structure (including any reusable or standard components and technologies), as well as design documents and models.

Build on a Scalable and Dependable Platform for Distributed Applications The Microsoft .NET Framework, along with Windows Server 2003, is designed to simplify application development in the highly distributed environment of the Internet and enterprise computing. This is achieved through integration of:

- * Public Internet standards such as XML, SOAP, UDDI, and WML.
- * Web services enhancements such as message-based security (WS-Security).
- * A highly scalable, loosely coupled architecture.

- * Application development in the language of your choice.
- * Easy-to-use automatic transactions, automatic memory management, and easy deployment.
- * Advanced security designed from the ground up to ensure that data and applications are protected through a fine-grained, evidence-based security model.
- * Rich operating system services such as transaction processing monitor and message queuing.

1.3.2 Enterprise Developer

Visual Studio .NET 2003 provides developers with powerful tools and servers for rapidly building XML Web services and enterprise applications. Visual Studio .NET 2003 Enterprise Developer builds on the power of Visual Studio .NET 2003 Professional by including additional capabilities for enterprise development teams building mission-critical applications that target any device and integrate with any platform. Developers using Visual Studio 2003 Enterprise Developer will benefit from:

A productive team development environment Development teams can safely version and share source code and documentation using Microsoft Visual SourceSafe®. Development guidelines and best practices can be shared across development teams using XML-based Template Description Language and Enterprise Template projects. Development teams can analyze performance and scalability of XML Web services and applications using Application Center Test (ACT).

The ability to build scalable and secure XML Web services and applications. Rapidly build, test, and deploy scalable and reliable enterprise applications that harness the power of servers, including full-featured developer versions of Microsoft Windows Server 2003 and Microsoft SQL Server™. Visually build server-side and database components using Server Explorer, Component Designer, and Visual Database Tools. Use the new Web Services Development Kit (WSDK) to support several core Web service scenarios, including message-based security (WS-Security).

A scalable, dependable platform for distributed applications. Create secure, reliable, high-performance applications using the Visual Studio .NET 2003 integrated development environment (IDE) and the Windows Server 2003 programming model. Choose from a broad range of integrated third-party tools.

Finally, it provides a powerful, enterprise team development environment for rapidly building mission-critical applications that target any device and integrate with any platform.

1.3.3 Professional

Visual Studio .NET 2003 Professional enables you to rapidly build a broad range of applications for Microsoft Windows®, the Web, and mobile devices. Developers can use Visual Studio .NET 2003 Professional to:

* ***Quickly build professional software.*** With an extensive set of visual designers, a range of programming languages, and integrated Visual Database Tools, Visual Studio .NET 2003 enables you to build powerful software quickly. Visual Studio .NET 2003 delivers the developer productivity you need to deliver a range of professional software in record time. The integrated development environment (IDE) provides a consistent interface for all languages, including Microsoft Visual Basic® .NET, Microsoft Visual C++® .NET, Microsoft Visual C#® .NET, and Microsoft Visual J#™ .NET. Using the language best suited to your skill set, you can take advantage of shared visual designers to build rich Windows-based applications and dynamic Web applications that render in any browser. Extra features are;

- Quickly Build Windows-based Applications
- Streamline Web-based Development
- Create Server-Side Business Logic
- IDE Productivity and Extensibility
- Build Software Using Your Choice of Modernized, Powerful Languages

* ***Reduce IT operating costs.*** Easy, Web-style deployment of rich Windows-based applications, built-in security, and an infrastructure for reusing existing code make the latest version of the Microsoft .NET Framework a dependable platform for software development. Visual Studio .NET 2003 contains an updated version of the .NET Framework, version 1.1, that builds upon the previous version with new capabilities and improved scalability, reliability, security, and performance. These core enhancements in the underlying development framework combine with improved IDE responsiveness to enable reduced costs associated with application development, deployment, and ongoing maintenance.

- Stability and Security
- Simplified Application Deployment
- Flexibility to Leverage Existing Investments

* ***Integrate with a wide range of applications, systems, and devices.*** Support for the latest XML Web service standards and visual designers for mobile application development enable you to easily extend your applications to other systems

and devices. Addressing the emerging opportunities represented by XML Web services and mobile application development were key design goals of Visual Studio .NET 2003. Using the skills you already have, Visual Studio .NET 2003 enables you to create and consume XML Web services and provides intuitive designers for rapidly building broad-reach Mobile Web applications as well as smart device software.

- XML Web Services
- Mobile Application Development

1.3.4 Academic

Visual Studio .NET 2003 Academic offers full support for the Microsoft .NET Framework version 1.1 and includes significant improvements for device development. Visual Studio .NET 2003 Academic also provides full support for the .NET Compact Framework, allowing seamless development for mobile and embedded devices, such as the Pocket PC and Pocket PC phone edition, as well as other devices powered by the Windows® CE .NET operating system. Web Services Enhancements (WSE) in Visual Studio .NET 2003 adds support for the latest Web services standards, including routing, attachment, and security. Visual Studio .NET 2003 Academic also includes an improved Visual Basic® .NET Upgrade Wizard, as well as C++ enhancements that lead to greater ANSI/ISO compliance. With Windows Server 2003 and Visual Studio .NET 2003 Academic, students can learn to rapidly develop and deploy dependable, connected applications.

Visual Studio .NET 2003 Academic brings the power of Microsoft .NET Framework into the classroom. It combines all of the features of Visual Studio .NET 2003 Professional with new tools and features, including Assignment Manager, student and faculty documentation, and sample code. All of these simplify course management by providing a set of tools that enable the publication of courses and assignments for students that may be accessed from a Web server, a network share, or a File Transfer Protocol (FTP) site. Visual Studio .NET 2003 Academic offers:

** A flexible and rich learning environment for student developers.* With the help of features like the object browser, class browser, and IntelliSense®, beginning students can master basic programming, and advanced students can learn to build complex XML Web services and applications. Students can also rapidly build Microsoft Windows applications for Web and Mobile Devices.

** Powerful course management tools for faculty.* Added functionality enables the publishing of courses and assignments to locations where students may access them using Visual Studio .NET 2003 Academic.

** Special features designed for academic needs.* Both students and faculty benefit from special features designed for instructional settings.

Learn to Program in a Rich and Flexible Environment ; Multiple-language support and interoperability enable students to write applications using a programming language of their choice. Visual Studio .NET 2003 Academic supports languages, that are used by more than 70 percent of all professional developers, including Visual Basic, C, C++, C#, and Java. In addition, the Microsoft .NET Framework has built-in support for dozens of other programming languages, including COBOL, Fortran, Scheme, Oberon, and Component Pascal. Cross-language inheritance enables students to share and reuse code across multiple languages, while cross-language debugging makes it easy to locate code errors in a multi-language environment.

Students can learn to quickly build powerful database, server, and Microsoft Windows-based graphical user interface (GUI) applications by employing XML, Windows Forms, Web Forms, and ADO.NET tools. Additionally, students can create solutions that work with any Internet-enabled device and integrate with any platform using XML services. ASP.NET mobile controls (formerly the Microsoft Mobile Internet Toolkit) and Smart Device Programmability in Visual Studio .NET 2003 Academic enable students to create a single Web solution that integrates smoothly with any operating system and mobile device, including mobile phones, Pocket PCs, handheld devices, and pagers.

Utilize Faculty Tools for Course Management Useful features such as Code Extraction help faculty create student starter projects by removing designated code from working code solutions. The starter project is then published with the assignment and provides the base for the student's coding work. This feature may be used with either the Assignment Publishing Tools or Assignment Manager. Assignment Publishing Tools enable faculty to publish courses and assignments so that students may access them from a Web server, a network share, or an FTP site. Once an assignment is published, the Assignment Manager tool helps instructors to submit assignments securely, track assignment submissions, automatically build student solutions, notify students of assignment grades, and send messages to students.

Facilitate Learning Through Special Instructional Features Visual Studio .NET 2003 Academic includes nine complete student assignments written in languages supported by the Visual Studio environment and ranging in difficulty from simple to advanced. This sample set of assignments includes those commonly found in computer science or information systems programs: Hello World; Tic-Tac-Toe; Diff Tool; Elementary Data Structures; Expression Parser; Sorting; Tile Puzzle; Towers of Hanoi; and Network Chat. The in-depth documentation in Visual Studio .NET 2003 Academic supports both faculty and student users. Students can learn quickly using detailed walkthroughs for topics, such as creating console applications and using the debugger. Faculty documentation describes the course management tools and provides instructions for deploying Visual Studio .NET 2003 Academic in computer science laboratories. The documentation also includes a rich Help reference.

Teach Server-Side Programming Quickly and Easily With the rapid application development (RAD) features in Visual Studio .NET 2003 Academic, faculty can teach server-side programming in first- and second-year computer science classes. For example, students can compose middle-tier components visually using the Visual Component Designer, which enables them to drag and drop message queues, timers, event logs, and other non-visual objects from the Toolbox. This reduces the amount of code required to build middle-tier components, while allowing for greater functionality in student assignments.

1.4 An Overview to C# Programming Language

1.4.1 Architectural History

C#'s principal designer, and lead architect at Microsoft, was Anders Hejlsberg. His previous experience in programming language and framework design (Visual J++, Borland Delphi, Turbo Pascal) can be readily seen in the syntax of the C# language, as well as throughout the CLR (Common Language Runtime) core. He can be cited in interviews and technical papers as stating flaws in most major programming languages, for example, C++, Java, Delphi, Smalltalk, were what drove the fundamentals of the CLR, which, in turn, drove the design of the C# programming language itself. His expertise can be seen in C#. There is a critical argument that C# shares roots in other languages, as purported by programming language history chart. C# was designed to fit both demands for a concise syntax (C++) and 'unlimited' rapid development (versus the 'limited' RAD of Visual Basic).

1.4.2 Object Oriented Programming

The idea behind object-oriented programming is that a computer program may be seen as comprising a collection of individual units, or *objects*, that act on each other, as opposed to a traditional view in which a program may be seen as a collection of functions, or simply as a list of instructions to the computer. Each object is capable of receiving messages, processing data, and sending messages to other objects. Each object can be viewed as an independent little machine or actor with a distinct role or responsibility.

Object-oriented programming is claimed to promote greater flexibility and maintainability in programming, and is widely popular in large-scale software engineering. Furthermore, proponents of OOP claim that OOP is easier to learn for those new to computer programming than previous approaches, and that the OOP approach is often simpler to develop and to maintain, lending itself to more direct analysis, coding, and understanding of complex situations and procedures than other programming methods. Critics dispute this, at least for some domains (industries).


Object-oriented programming (OOP) emphasizes the following concepts:


- * Class — the unit of definition of data and behavior (functionality) for some kind-of-thing. For example, the 'class of Dogs' might be a set which includes the various breeds of dogs. A class is the basis of modularity and structure in an object-oriented computer program. *A class should typically be recognizable to a non-programmer familiar with the problem domain*, and the code for a class should be (relatively) self-contained and independent (as should the code for any good pre-OOP function). With such modularity, the structure of a program will correspond to the aspects of the problem that the program is intended to solve. This simplifies the mapping to and from the problem and program.

- * Object — an instance of a class, an object (for example, "Lassie" the Dog) is the run-time manifestation (*instantiation*) of a *particular exemplar* of a class. (For the class of dogs which contains breed types, an acceptable exemplar would only be the *subclass* 'collie'; "Lassie" would then be an object in that subclass.) Each object has its own data, though the code within a class (or a subclass or an object) may be shared for economy. .

- * Method (also known as *message*) — how code can use an object of some class. A *method* is a form of subroutine operating on a single object. Methods may be divided into queries returning the current state and commands changing it: a Dog could have a


query Age to say how old it is, and command chase (Rabbit target) to start it chasing a rabbit. A method may also do both, but some authorities (e.g. Bertrand Meyer) recommend they be kept separate. Sometimes access to the data of an object is restricted to the methods of its class.


 *A member of a class or object is a method or a data item describing the state of an object. In some languages the general term is feature.

 * Inheritance — a mechanism for creating subclasses, inheritance provides a way to define a (sub)class as a specialization or subtype or extension of a more general class: Dog is a subclass of Canidae, and Collie is a subclass of the (sub)class Dog. A subclass *inherits* all the members of its superclass(es), but it can extend their behaviour and add new members. Inheritance is the "is-a" relationship: a Dog *is a* Canidae. This is in contrast to composition, the "has-a" relationship: a Dog *has a* mother (another Dog) and *has a* father, etc.

* Multiple inheritance – a Dog is both a Pet and a Canidae – is not always supported, as it can be hard both to implement and to use well.

* Encapsulation — ensuring that code outside a class sees only functional details of that class, but not implementation details. The latter are liable to change, and could allow a user to put an object in an inappropriate state. Encapsulation is achieved by specifying which classes may use the members of an object. The result is that each object exposes to any class a certain *interface* — those members accessible to that class. For example, an interface can ensure that puppies can only be added to an object of the class Dog by code in that class. Members are often specified as public, protected and private, determining whether they are available to all classes, sub-classes or only the defining class. Some languages go further: Java uses the protected keyword to restrict access also to classes in the same package, C# and VB.NET reserve some members to classes in the same assembly using keywords internal (C#) or Friend (VB.NET), and Eiffel allows one to specify which classes may access any member.

 * Abstraction — the ability of a program to ignore the details of an object's (sub)class and work at a more generic level when appropriate; For example, "Lassie" the Dog may be treated as a Dog much of the time, but when appropriate she is abstracted to the level of Canidae (superclass of Dog) or Carnivora (superclass of Canidae), and so on.

 * Polymorphism — polymorphism is behavior that varies depending on the class in which the behavior is invoked, that is, two or more classes can react *differently* to the

same message. For example, if Dog is commanded to speak this may elicit a Bark; if Pig is commanded to speak this may elicit a Grunt.

1.4.3 User Defined Classes in C#

In C#, most of object definition, class methods and syntax has been taken from C and C++. So, to create a class in C# we use “class” keyword and a class name. Public, private or protected attributes of the class determined via these keywords. To write methods for variable’s get and set operations are implemented very easily. The keywords “get” and “set” can be used to create this method. An example from project is shown below:

```
using System;
```

```
namespace Graduation2
```

```
{  
    public class Customer  
    {  
        private string sUserEMail;  
        private int iRestaurantID;  
        public string UserEMail  
        {  
            get{return sUserEMail;}  
            set{sUserEMail = value;}  
        }  
        public Customer(){}  
    }  
}
```

In the example code above, “using” keyword is used as “include” in C. System header file is the main one for creating a class. The keyword “namespace” is used to determine the solution name we use class in. The method Customer () is the constructor of the class. It is called automatically when a component from this class is created.

1.4.4 Language Features

C# is, in some senses, the programming language which most directly reflects the underlying Common Language Infrastructure (CLI). It depends strongly on this framework because it was designed specifically to take advantage of the features that the CLI provides. Most of C#'s intrinsic types correspond to value-types implemented by the CLI framework. A common misbelief is that they are garbage-collected, though

they are not; they are true value-types and are stack allocated (with an exception for `System.Object`, and due to interning, `System.String`). Applications written in C# require an implementation of the Common Language Runtime (CLR) to execute, in the same way that VB6 requires a runtime to execute (this is often confused with the JRE, which provides a byte-code interpreter). Unlike Java classes, CLI applications are compiled in 2 passes. First compiled to platform abstract bytecode, and secondly, it's compiled at first runtime of the application to the native client's machine code. Compared to C and C++, the language is restricted or enhanced in a number of ways, including but not limited to the following:

- * True support for pointers. However pointers can only be used within *unsafe* scopes, and only programs with appropriate permissions can execute code marked as unsafe. Most object access is done through safe references, which cannot be made invalid, and most arithmetic is checked for overflow. An unsafe pointer can be made to not only value-types, but to subclasses of `System.Object` as well. Also safe code can be written that uses a pointer (`System.IntPtr`).

- * Managed memory cannot be explicitly freed, but instead is garbage collected when no more references to the memory exist. (Objects that reference unmanaged resources, such as an `HBRUSH`, can be instructed to release those resources through the standard `IDisposable` interface, which provides a pattern for deterministic deallocation of resources.)

- * Multiple inheritance is prohibited (although a class can implement any number of interfaces). This was a design decision by the language's lead architect (Anders Hejlsberg) to avoid complication, avoid 'dependency hell,' and simplify architectural requirements throughout CLI.

- * C# is more typesafe than C++. The only implicit conversions by default are safe conversions, such as widening of integers and conversion from a derived type to a base type (and this is enforced at compile-time and, indirectly, during JIT). There are no implicit conversions between booleans and integers and between enumeration members and integers, and any user-defined implicit conversion must be explicitly marked as such, unlike C++'s copy constructors.

- * Syntax for array declaration is different (`"int[] a = new int[5];"` instead of `"int a[5];"`).

- * Enumeration members are placed in their own namespace.

- * C# 1.0 lacks templates, however, C# 2.0 provides generics.

- * Properties are available which results in syntax that resembles C++ member field access, similar to VB.

- * Full type reflection and discovery is available.

1.4.5 C# 2.0 & 3.0 New Language Features

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

- * Partial classes (separation of class implementation into more than one file)

Generics or parameterized types. They support some features not supported by C++ templates such as type constraints on generic parameters. On the other hand, expressions cannot be used as generic parameters, as with C++ templates. Also, they differ from Java in that parameterized types are first-class objects in the Virtual Machine, which allows for optimizations and preservation of the type information. See a simple example of C# 2.0 generics.

- * Static classes which represent a concept close to VB.NET modules, cannot be instantiated from code and allow only static members.

- * A new form of iterator that employs coroutines via a functional-style `yield` keyword similar to `yield` in Python.

- * Anonymous delegates providing closure functionality.

- * Covariance and contravariance for signatures of delegates

- * Coalesce operator: (`??`) returns the first non-null value in a list:

```
object nullObj = null; object obj = new Object();  
return nullObj ?? obj; //returns obj
```

The primary use of this operator is to assign a nullable type to a non-nullable type with an easy syntax:

```
int? i = null;  
int j = i ?? default(int); //can't assign null to int
```

In C# 3.0 there will be radical additions:

- * "select, from, where" keywords allowing to query from SQL, XML, collections, and more (LINQ).

- * Object initialization : `Customer c = new Customer(); c.Name="James";` becomes `Customer c = new Customer { Name="James" };`

- * Lambda expressions : `listOfFoo.Where(delegate(Foo x) { return x.size>10;})` becomes `listOfFoo.Where(x => x.size>10);`

- * Local variable type inference: `var x = "hello";` is interchangeable with


```
string x="hello";
```

```
* Anonymous types : var x = new { Name = "James" }
```

```
* Extension methods (adding methods to classes by including the this keyword in the first parameter)
```

C# 3.0 was unveiled at the PDC 2005, and a Preview, with specifications is available From the MSDN Page (MSDN).

Language researchers at Microsoft have emphasized that C# 3.0 is bytecode-compatible with C# 2.0 — essentially the improvements are purely syntactic or compile-time improvements. For example, many of the most common integrated queries can already be implemented using anonymous delegates in combination with predicate-based container methods such as List.FindAll and List.RemoveAll.

Code Libraries ; The ECMA C# specification details a minimum set of types and class libraries that the compiler expects to have available and they define the basics required. Most implementations in the open ship with the larger set of libraries.

The .NET Framework is a class library which can be used from a .NET language to perform tasks from simple data representation and string manipulation to generating dynamic web pages (ASP.NET), XML parsing, Web Services/Remoting (SOAP) and reflection. The code is organized into a set of namespaces which group together classes with a similar function, e.g. System.Drawing for graphics, System.Collections for data structures and System.Windows.Forms for the Windows Forms system.

A further level of organisation is provided by the concept of an *assembly*. An assembly can be a single file or multiple files linked together (through al.exe) which may contain many namespaces and objects. Programs needing classes to perform a particular function might reference assemblies such as System.Drawing.dll and System.Windows.Forms.dll as well as the core library (known as mscorlib.dll in Microsoft's implementation).

Standartization ; In August, 2000, Microsoft Corporation, Hewlett-Packard and Intel Corporation co-sponsored the submission of specifications for C# as well as the Common Language Infrastructure (CLI) the international standardization organization ECMA. In December 2001, ECMA released ECMA-334 *C# Language Specification*. C# became an ISO standard in 2003 (ISO/IEC 23270). ECMA had previously adopted equivalent specifications as the 2nd edition of C#, in December, 2002.

In June 2005, ECMA approved edition 3 of the C# specification, and updated ECMA-334. Additions included partial classes, anonymous methods, nullable types,

and generics (similar to C++ templates). In July 2005, ECMA submitted the standards and related TRs to ISO/IEC JTC 1 via the latter's Fast-Track process. This process usually takes 6-9 months. Based on these standards, there are independent implementations being worked on, including:

- * Mono, Novell's open source .NET implementation (originally by Ximian).
- * DotGNU, and Portable.NET from the Southern Storm Software, PTY

Microsoft released support of the 3rd edition of C# in the .NET SDK 2.0, and Visual Studio 2005, in November 2005.

Politics ; Many of Microsoft's products and initiatives generate political attention, and C# is no exception. Owing to C#'s close relationship with a commercial institution, political discussions continue regarding the legitimacy of C# standardization, its Java similarities, its future as a general-purpose language, and other issues. Some security experts express skepticism as to the efficacy of the CLR's security mechanisms, and criticise their complexity. At the same time, the language is praised for its clear and programmer-friendly grammar, in addition to reduction in development time for certain types of applications. Unlike proprietary languages such as Visual Basic, Microsoft chose to open up C# to the standardization process. However, Microsoft is still a primary force driving changes and innovation in the language. Additionally, Microsoft has made it clear that C#, as well as the other .NET languages, is an important part of its software strategy for both internal use and external consumption. Microsoft takes an active role in marketing the language as part of its overall business strategies.

Language Name; According to the ECMA-334 C# Language Specification, section 6, Acronyms and abbreviations [1] the name of the language is written "C#" ("LATIN CAPITAL LETTER C (U+0043) followed by the NUMBER SIGN # (U+0023)") and pronounced "C Sharp".

C sharp musical note Due to technical limitations of display (fonts, browsers, etc.) and the fact that the sharp symbol (#, U+266F, MUSIC SHARP SIGN, see graphic at right if the symbol is not



visible) is not present on the standard keyboard, the number sign (#) was chosen to represent the sharp symbol in the written name of the language. So, although the symbol in "C#" represents the sharp symbol, it is actually the number sign ("#"). Although Microsoft's C# FAQ refers to the sharp symbol in the language name, Microsoft clarifies the language name as follows:

"The spoken name of the language is "C sharp" in reference to the musical "sharp" sign, which increases a tone denoted by a letter (between A and G) by half a tone. However, for ease of typing it was decided to represent the sharp sign by a pound symbol (which is on any keyboard) rather than the "musically correct" Unicode sharp sign. The Microsoft and ECMA 334 representation symbols thus agree: the # in C# is the pound sign, but it represents a sharp sign. Think of it in the same way as the <= glyph in C languages which is a less than sign and an equals sign, but represents a less-than-or-equals sign.", Microsoft Online Customer Service.

The choice to represent the sharp symbol (#) with the number sign (#) has led to confusion regarding the name of the language. For example, although most printed literature uses the correct number sign [2], some incorrectly uses the sharp symbol. What's more, users have been known to call the language "see-pound" (in the US the #-key on telephones is pronounced as the "pound"-key) or "see-hash". Also in the US the # symbol is also occasionally referred to as the "gate" symbol on a telephone, leading to a pronunciation of the language as "see-gate", which could be confused with the brand name of hard-drive manufacturer, Seagate.

The "sharp" suffix has been emulated by a number of other .NET languages that are variants of existing languages, including J# (Microsoft's implementation of Java), A# (from Ada), F# (presumably from System F, the type system used by the ML family), and Gtk# (a .NET wrapper for GTK+).

The # symbol also makes a visual programmers joke. The ++ (increment) operator signifies an increase, or improvement. Hence C++ is an improvement over C. C# can be seen as ++ stacked on top of ++, therefore even a further increase.

1.5 Overview of SQL Server Developer Edition

1.5.1 Understanding the Definition of Database

Both SQL and relational database theory originated in IBM's research laboratories. In June 1970, Dr. Edgar F. Codd, an IBM engineer, wrote a paper outlining the mathematical theory of how data could be stored in tables and manipulated using a data sublanguage. The article, entitled "A Relational Model of Data for Large Shared Data Banks," was published in the Communications of the Association for Computing Machinery (ACM) and led to the creation of relational database management systems (DBMS) and Structured Query Language (SQL).

After Dr. Codd published his article, IBM researchers began work on System /R, a prototype relational DBMS. During the development of System /R, the engineers also worked on a database query language-after all, once data was stored in a DBMS, it would be of no use unless you could combine and extract it in the form of useful information. One of the query languages, SEQUEL (short for Structured English Query Language), became the de facto standard data query language for relational DBMS products. The SQL we use today is the direct descendant of IBM's original SEQUEL data sublanguage.

Although IBM started the research in 1970 and developed the first prototype relational DBMS (System /R) in 1978, it was Oracle (then known as Relational Software, Inc.) that introduced the first commercial relational DBMS product in 1980. The Oracle DBMS (which ran on Digital Equipment Corp [DEC] VAX minicomputers) beat IBM's first commercial DBMS product (SQL/DS) to market by two years. While Oracle continued to refine its product and released version 3, which ran on mainframes, minicomputers, and PCs, in 1982, IBM was working on Database 2 (DB2) which it announced in 1983 and began shipping in 1985. DB2 operated on IBM's MVS operating system on IBM mainframes that dominated the large data center market at the time. IBM called DB2 its flagship relational DBMS, and with IBM's weight behind it, DB2's SQL became the de facto standard database language. Although initially slower than other database models (such as the hierarchical model that you learned about in Tip 3, "Understanding the Hierarchical Database Model," and the network model that you learned about in Tip 4, "Understanding the Network Database Model"), the relational model had one major advantage-you didn't need a programmer to get information from the database. The relational query languages let users pose ad hoc, English-like queries to the database and get immediate answers-without having to write a program first. As the performance of relational DBMS products improved through software enhancements and increases in hardware processing power, they became accepted as the database technology of the future. Unfortunately, compatibility across vendor platforms was poor. In 1986 the American National Standards Institute (ANSI) and the International Organization for Standardization (ISO) published the first formal ANSI/ISO standard for SQL. SQL-86 (or SQL1) gave SQL "official" status as *the* relational DBMS data language. ANSI updated the standard in 1992 to include "popular" enhancements/extensions found across DBMS products and added a "wish list" objects and methods that a DBMS *should* have.

SQL-92 (or SQL2), published in ANSI Document X3.135-1992, is the most current and comprehensive definition of SQL. At present, no commercial DBMS fully supports all of the features defined by SQL-92, but all vendors are working toward becoming increasingly compliant with the standard. As a result, we are getting closer to the goal of having a data language (SQL) that is truly transportable across DBMS products and hardware platforms.

Many people use the term database to mean any collection of data items. Microsoft and Lotus have also blurred the lines between application data and a database by referring to "database" queries in help screens about searching the information stored in the cells that make up their competing spreadsheet products.

As the name implies, a database contains data. The data is organized into records that describe a physical or conceptual object. Related database records are grouped together into tables. A customer record, for example, could consist of data items, or attributes, such as name, customer number, address, phone number, credit rating, birthday, anniversary, and so on. In short, a customer record is any group of attributes or characteristics that uniquely identify a person (or other business), making it possible to market the customer for new business or to deliver goods or services. A customer table, then, is a collection of customer records. Similarly, if a business wants to track its inventory (or collection of goods for sale), it would create an inventory table consisting of inventory records. Each inventory record would contain multiple attributes that uniquely describe each item in the inventory. These attributes might include item number, description, cost, date manufactured or purchased, and so on.

1.5.2 Database Architecture And Features of SQL 2000

Microsoft® SQL Server™ 2000 data is stored in databases. The data in a database is organized into the logical components visible to users. A database is also physically implemented as two or more files on disk.

When using a database, you work primarily with the logical components such as tables, views, procedures, and users. The physical implementation of files is largely transparent. Typically, only the database administrator needs to work with the physical implementation.

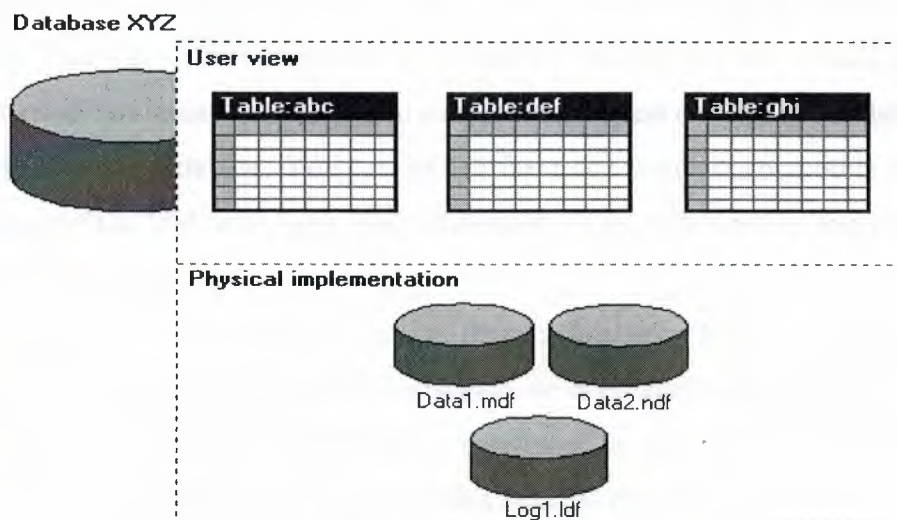


Figure 1.1 Database Models

Each instance of SQL Server has four system databases (**master**, **model**, **tempdb**, and **msdb**) and one or more user databases. Some organizations have only one user database, containing all the data for their organization. Some organizations have *different databases for each group in their organization, and sometimes a database used by a single application*. For example, an organization could have one database for sales, one for a document management application, and so on. Sometimes an organization uses only one database; other applications may access several databases.

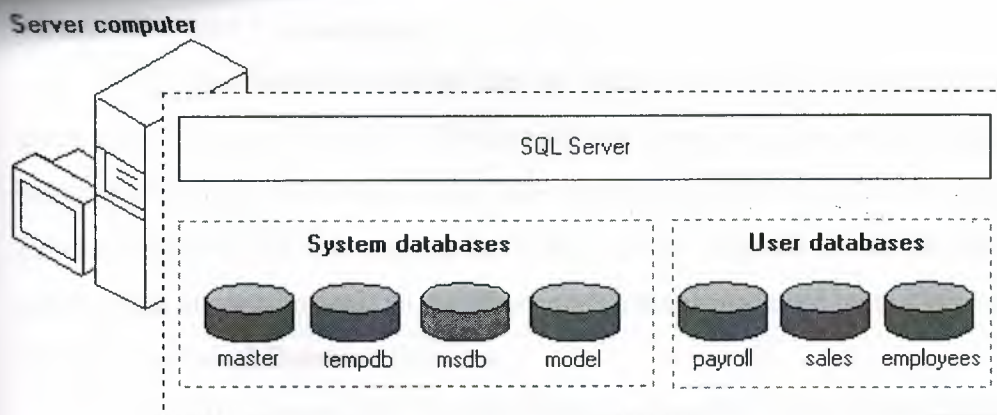


Figure 1.2 Database Tables

It is not necessary to run multiple copies of the SQL Server database engine to allow multiple users to access the databases on a server. An instance of the SQL Server Standard or Enterprise Edition is capable of handling thousands of users working in multiple databases at the same time. Each instance of SQL Server makes all databases in the instance available to all users that connect to the instance, subject to the defined security permissions. When connecting to an instance of SQL Server, your connection is

associated with a particular database on the server. This database is called the current database. You are usually connected to a database defined as your default database by the system administrator, although you can use connection options in the database APIs to specify another database. You can switch from one database to another using either the Transact-SQL *USE database_name* statement, or an API function that changes your current database context.

SQL Server 2000 allows you to detach databases from an instance of SQL Server, then reattach them to another instance, or even attach the database back to the same instance. If you have a SQL Server database file, you can tell SQL Server when you connect to attach that database file with a specific database name.

Microsoft® SQL Server™ 2000 features include:

*** Internet Integration :**

The SQL Server 2000 database engine includes integrated XML support. It also has the scalability, availability, and security features required to operate as the data storage component of the largest Web sites. The SQL Server 2000 programming model is integrated with the Windows DNA architecture for developing Web applications, and SQL Server 2000 supports features such as English Query and the Microsoft Search Service to incorporate user-friendly queries and powerful search capabilities in Web applications.

*** Scalability and Availability:**

The same database engine can be used across platforms ranging from laptop computers running Microsoft Windows® 98 through large, multiprocessor servers running Microsoft Windows 2000 Data Center Edition. SQL Server 2000 Enterprise Edition supports features such as federated servers, indexed views, and large memory support that allow it to scale to the performance levels required by the largest Web sites.

*** Enterprise-Level Database Features:**

The SQL Server 2000 relational database engine supports the features required to support demanding data processing environments. The database engine protects data integrity while minimizing the overhead of managing thousands of users concurrently modifying the database. SQL Server 2000 distributed queries allow you to reference data from multiple sources as if it were a part of a SQL Server 2000 database, while at the same time, the distributed transaction support protects the integrity of any updates of the distributed data. Replication allows you to also maintain multiple copies of data, while ensuring that the separate copies remain synchronized. You can replicate a set of

data to multiple, mobile, disconnected users, have them work autonomously, and then merge their modifications back to the publisher.

*** Ease of installation, deployment, and use :**

SQL Server 2000 includes a set of administrative and development tools that improve upon the process of installing, deploying, managing, and using SQL Server across several sites. SQL Server 2000 also supports a standards-based programming model integrated with the Windows DNA, making the use of SQL Server databases and data warehouses a seamless part of building powerful and scalable systems. These features allow you to rapidly deliver SQL Server applications that customers can implement with a minimum of installation and administrative overhead.

*** Data warehousing :**

SQL Server 2000 includes tools for extracting and analyzing summary data for online analytical processing. SQL Server also includes tools for visually designing databases and analyzing data using English-based questions.

1.5.3 Ms-SQL Server Permissions and Uses

Permissions, or privileges, are the rights a user ID has to access a database object such as a table, view, domain, or stored procedure. The database administrator (DBA) controls user and application program interaction with the database by granting and revoking privileges (permissions) to user IDs and roles.

In addition to permissions *explicitly* granted by the DBA, the DBMS also *implicitly* (automatically) gives the full set of object privileges to the owner or creator of an object. As such, the database object owner (DBOO) automatically has SELECT, INSERT, UPDATE, DELETE, EXEC, DIR (foreign key references), and data definition language (DDL) privileges on the database objects the user or application program creates. Moreover, the DBOO can also GRANT and REVOKE privileges to other user IDs.

The privileges, or permissions, available for database objects on MS-SQL Server are:

SELECT. Lets a user-ID query or read data from a table or view. The SELECT privilege can be limited to individual columns within a table or view, or granted on the entire object as a whole.

INSERT. Lets a user ID add rows to a table or view.

UPDATE. Lets a user ID change data in a table or view. Similar to the SELECT privilege, the UPDATE privilege can be limited to individual columns within a table or view or granted on the entire object as a whole.

DELETE. Lets a user ID remove rows from a table or view.

EXEC. (Execute.) Lets a user ID execute a stored procedure.

DRI. (Declarative referential integrity.) Lets a user ID add FOREIGN KEY constraints on a table.

CREATE TABLE. Lets the user ID add a new table to the database. As you already know, the user ID will become the DBOO of the table it creates and thus will have all privileges on that table.

ALTER TABLE. Lets the user ID change the structure of a table.

DROP TABLE. Lets the user ID DROP or delete a table.

CREATE. Lets the user ID create the database object (such as a DEFAULT, PROCEDURE, RULE, or VIEW) to which it is granted the ALTER privilege. As was mentioned previously, the user ID will become the DBOO of the database object it creates, with all permissions (privileges) on the new database object.

ALTER. Lets the user ID modify the structure of or the statements in (in the case of stored procedures) the database object (such as a DEFAULT, PROCEDURE, RULE, or VIEW) to which it is granted the ALTER privilege.

DROP. Lets the user ID delete or remove the database object (such as a DEFAULT, PROCEDURE, RULE, or VIEW) to which it is granted the DROP privilege.

BACKUP DATABASE. Lets the user ID back up an entire database to a backup device.

BACKUP LOG. Lets the user ID back up a database transaction log to a backup device.

CREATE DATABASE. Lets the user ID create a new database on the SQL Server. After creating the database, the user ID will become the DBO for the database and thus will have all permissions on all objects created in the new database by any user ID.

When using SQL to send commands to the DBMS, you first tell the DBMS what you want to do and then describe the data (or structure) on which you want the DBMS to take the action. SQL is similar to the German language in that you put the action word (the verb) at the beginning of the sentence (the SQL statement) and then follow the verb with one or more clauses that describe the subject (the database object, or set of rows) on which you want the DBMS to act. Figure shows the basic form of SQL statements.

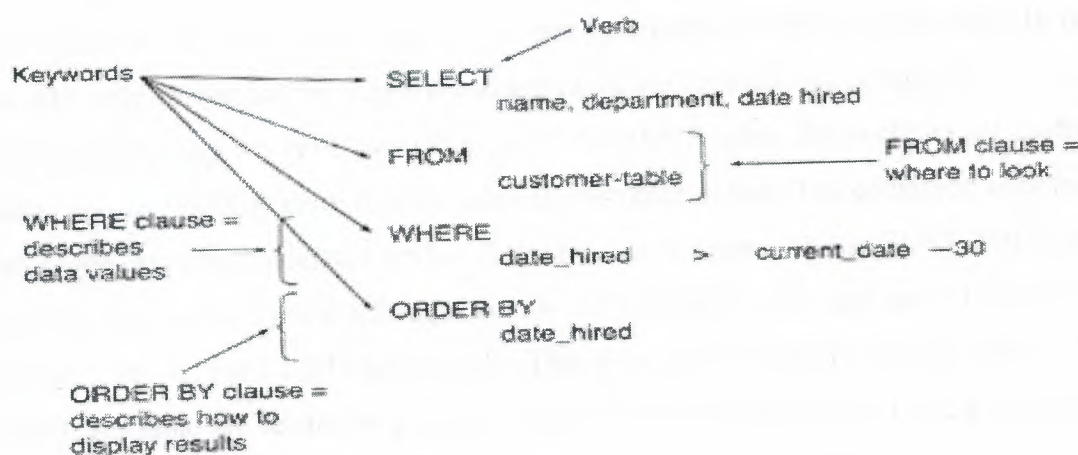


Figure 1.3 Statements Rule

SELECT

NAME, ADDRESS, CITY, STATE, ZIP, PHONE_NUMBER,
BALANCE_DUE

FROM CUSTOMER

WHERE BALANCE_DUE > 1000.00 ORDER BY ZIP ;

Database Objects; **Keys**

There are two types of keys: Primary and foreign. Both of them can be created when you create the table or after the fact, as long as you don't violate any of the rules. *Primary Key* is used to unique identify each row on a table. It can either be part of the actual record itself, or it can be an artificial field (something that has nothing to do with the actual record). A primary key can consists of one or more fields on a table. When multiple fields are used as a primary key, it is called a composite key.

Primary keys can be specified either when the table is created (using CREATE TABLE) or by changing the existing table structure (using ALTER TABLE).

Below are examples for specifying a primary key when creating a table:

MySQL:

```
CREATE TABLE Customer
(SID integer,
Last_Name varchar(30),
First_Name varchar(30),
PRIMARY KEY (SID));
```

Oracle:

```
CREATE TABLE Customer
(SID integer PRIMARY KEY,
Last_Name varchar(30),
First_Name varchar(30));
```

A *foreign key* is a field (or fields) that points to the primary key of another table. The purpose of the foreign key is to ensure referential integrity of the data. In other words, only values that are supposed to appear in the database are permitted. For example, say we have two tables, a CUSTOMER table that includes all customer data, and an ORDER table that includes all customer orders. The constraint here is that all orders must be associated with a customer that is already in the CUSTOMER table. In this case, we will place a foreign key on the ORDERS table and have it relate to the primary key of the CUSTOMER table. This way, we can ensure that all orders in the ORDERS table are related to a customer in the CUSTOMER table. In other words, the ORDERS table cannot contain information on a customer that is not in the CUSTOMER table.

The structure of these two tables will be as follows:

Table **Customer**

column name	characteristic
SID	Primary Key
Last_Name	
First Name	

Table **Orders**

column name	characteristic
Order_ID	Primary Key
Order_Date	
Customer SID	Foreign Key
Amount	

In the above example, the Customer SID column in the Order table is a foreign key pointing to the SID column in the Customer table. Below we show examples of how to specify the foreign key when creating the ORDER table:

MySQL:

```
CREATE TABLE Order
```

```
(Order_ID integer,
```

```
Order_Date date,
```

```
Customer_SID integer,
```

```
Amount double,
```

```
Primary Key (Order_ID),
```

```
Foreign Key (Customer_SID) references Customer(SID));
```


CHAPTER 2

PROJECT ANALYSIS & OVERVIEW

Program analysis has been increasingly used in software & computer engineering tasks such as auditing programs for security vulnerabilities and finding errors in general. In this chapter we are going to explain the analysis period and the structure period of the program. Also we will discuss about the parts of program. First of all we would like to define a few meanings of analysis in our daily life;

Analysis is a systematic approach to problem solving. Complex problems are made simpler by separating them into more understandable elements. This involves the identification of purposes and facts, the statement of defensible assumptions, and the formulation of conclusions.

Analysis, in philosophy, is principally an account of either the meaning or content of a word, phrase, or concept, and it may be applied to the analysis of an argument that makes use of such an analysis. In practice, analyses of words and concepts are not easily distinguishable from definitions. It is held by many contemporary philosophers that analyses, per se, are not possible; other terms used for the same sort of item are explication and account.

An analysis conducted to evaluate an installation's disposal decisions in terms of the environmental impact. The NEPA analysis is useful to the community's planning efforts and the installation's property disposal decisions. It is used to support DoD decisions on transferring property for community reuse. ...etc.

As we see there are much more meanings of analysis but all definitions are combined in one point that is separating a big thing to smaller parts and start to investigate it from the small parts.

In our project we divide complete program into sub portions. We had two main parts and an administration part. Cargo, transportation and admin part. For cargo portion we have a staff part named as cargostaff, manager part named as cargoman, for transportation portion also we have a staff part named as transstaff, a manager part named as transman. Program includes two more departments as personnel named as empman and account named as accountman. All these departments and sub portions will be clarified in this chapter as their works, permissions, and the main positions in the program. But before all these we want to show all these in a diagram to understand the whole departments in sgur company.

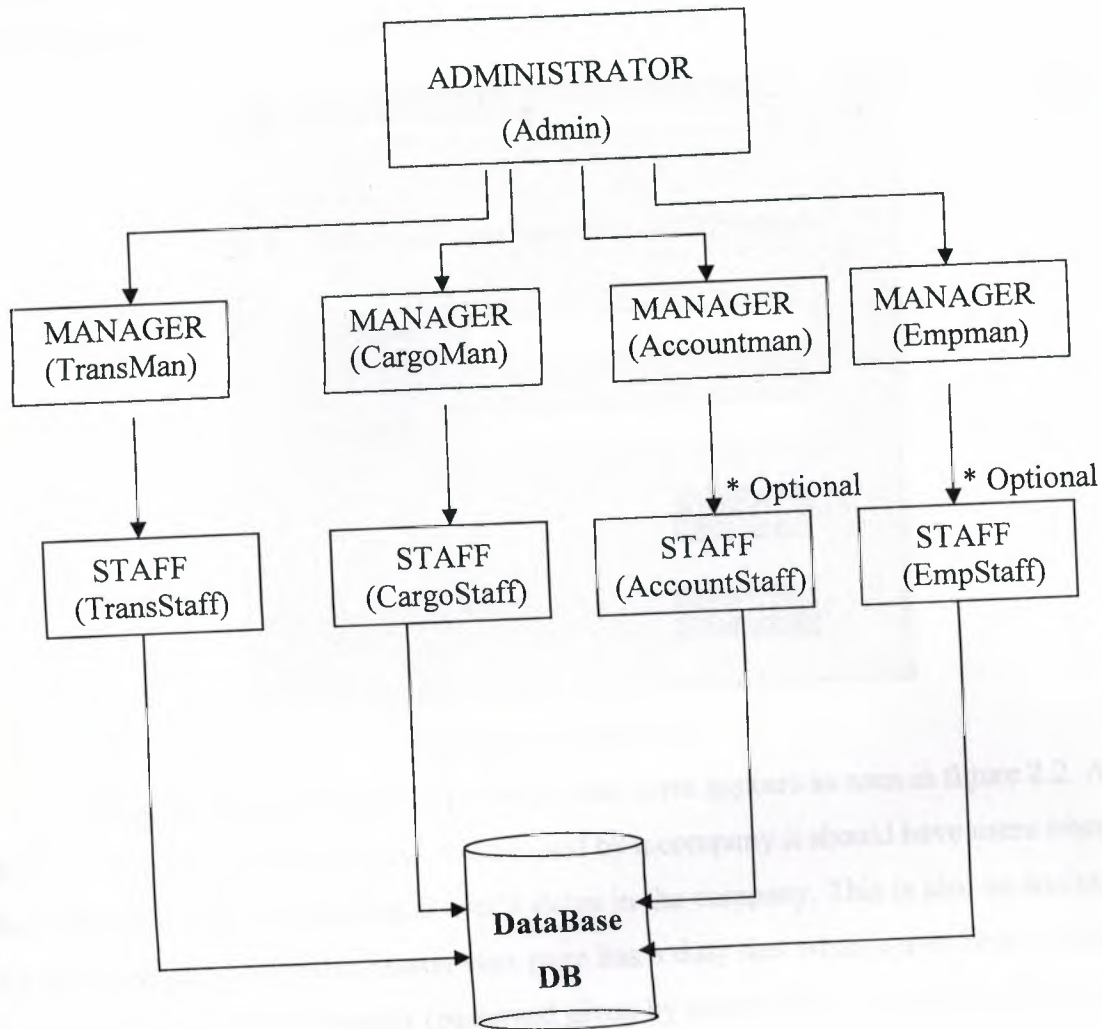


Figure 2.1 Departments and Users

2.1 Project Overview

We state for Accountstaff and EmpStaff as optional because our program is not include these two portions because for account and personel departments only one person is responsible for each but if it wants us can easily create staff portions for them too. For this reason we will not cover these two portions.

As we see from graph we have 7 main portions in our project. We said that in our analysis part we start with dividing whole departments in sub portions as in graph. This was the first analysis that we made, the second, we divide all these sub portions in itself due to responsibility of each department. We will discuss it now but before discussing all these we want to tell the third analysis part. Third part is the most important and efficient part because here we analysed the departments according to their works and responsibilities. For example what are the responsibilities of a cargostaff and

what do a cargo company do? Also transport department's too. So now we can start to investigate each portion of project starting from start form;

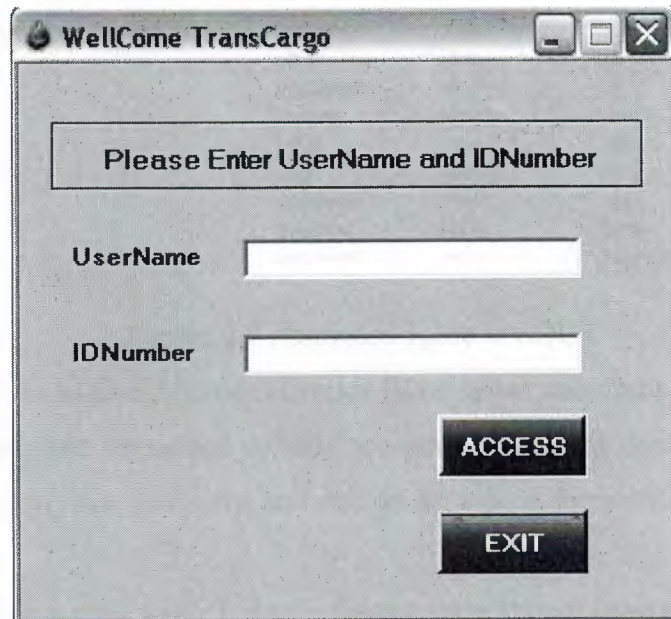


Figure 2.2 Start Form

When we start to debug our program, start form appears as seen in figure 2.2. As this is an application program and will be used by a company it should have users where the forms will appear according to user's status in the company. This is also an analysis for the company. So for this reason start page has a duty that when a user he/she enters his/her username and Id number (password given by admin of the program) a form will appear according to his/her permission and department. For example a worker in the company named Süleyman has user name "sülo" and Idnumber "12345"(when a user enters password he/she will see star icon for each number of his/her Idnumber for the security of program that the password will not seen from someone else), he has a permission of "3" and department is "cargo" means cargostaff form will appear for süleyman user.

This is the main idea for the start form but the problem is how it separates the permission and the department of the user? Now we are going to examine this.

We said that the username and Idnumber is given by the Administrator of the program. When the admin gives these, he/she has to fill seven datas in the Password table in SQL for that user. These are username, idnumber, permission, department, name, surname, personelid.

From these datas he can decide his/her permission, department and the program automatically guide the user to the form that he/she is responsible. Here we will not give

information about the codes that how program automatically do application, this will cover in the next chapter detailed.

	username	idnumber	permission	department	name	surname	personelid
	ad	111	1	admin	sinan	öz	1232333324
	ad1	1987	2	employee	ali	ali	5678443453
	as	888	3	account	Mesut	arik	7777773327
	cm	222	2	cargo	ozan	akkoca	6666345666
	cs	444	3	cargo	göktuğ	ataç	5553465555
	em	666	2	employee	engin	alan	4423423444
	tm	333	2	transport	ismail	hakki	2213257822
	ts	555	3	transport	sinan	özerenler	1134555411

Figure 2.3 Password Table in SQL

As we see from Figure 2.3 administrator filled seven columns for each user in the system. But the most important columns are permission and department because program will look these two columns and decide the which form will appear for user as we said.

Actually, when a user wants to login the program should have an username and Idnumber otherwise cannot login the system. So when a recorded user wants to login system first of all he/she will type the username then the Idnumber and will press the ACCESS button on the form. If user typed username or password wrong system will automatically warn the user that he/she had typed wrong and will want from user to try it again. Also if user enters username or password and will leave a textbox empty then try to push the ACCESS button then system again will automatically warn user that he/she has to fill all required data.

After typing the correct username and password to textboxes (or let's say the username and password which is exist in database) and push the ACCESS button program first goes to database table (figure 2.3) and looks is there a user whose username and Idnumber is like typed in textboxes. If the match is true then the program will take user's permission and department from selected row and opens the new form according to these information.

We said before that in this chapter we will not give information about codes, all these code behinds will be in next chapter. Actually the main logic is that we order the program that go and look the selected user's permission and department and with "IF" conditions open the desired form. We determined the permissions in the company as; Administrator permission → 1 Manager's permission → 2 Staff's permission → 3 . So as we gave an example that a worker wants to login system and his permission is "3" and his department is "cargo" the program will execute the code and automatically

understand that this user is a staff in cargo department and open the cargostaff form for user. As we start to talk about cargostaff form so now let's examine this form with respect to our topic as how we analysed cargostaff form and the contents of form. First have a look the appearance of the cargostaff form.

Figure 2.4 Cargostaff Form

The screenshot shows the 'CALCULATE PRICE' window of the Cargostaff application. At the top, there is a menu bar with icons for Search Cargo, Arrival Time, Calculate Price, Branch Info, New Cargo, Arrival Cargo, and HELP. On the left, the text 'S G C A R G O' is displayed vertically. On the right, there is a vertical menu with buttons for MENU, ADMIN, CARGO MANAGER, TRANS STAFF, CALC, and EXIT. The main area contains a form titled 'PLEASE FILL ALL REQUIRED INFORMATION TO CALCULATE PRICE'. The form has several input fields: Sender City (Izmir), Receiver City (Ankara), Send Type (Cargo), Send By (Normal), Weight (30-60 kg), Volume (85), Width (34 cm), Height (76 cm), and Altitude (99 cm). There is a 'Calculate' button at the bottom right. Below the form, there is a summary table showing the calculated price.

SENDER CITY		PRICE	
SENDER CITY	Izmir	30 YTL	
RECEIVER CITY	Ankara	INSURANCE	✓ (6YTL)
SEND TYPE	Cargo	SEND BY	Normal
WEIGHT	30-60	TAX (%18)	5,4
VOLUME	85cm*3 *(4YTL)	TOTAL PRICE	45,4 YTL
DIMENSION	34-76-99		

Figure 2.4 Cargostaff Form

Cargostaff as seen in the figure includes 7 main parts ;

- Search Cargo
- Arrival Time
- Calculate Price
- Branch Info
- New Cargo
- Arrival Cargo
- Help

As we see we have 7 main buttons in the upper part of form and have five sub buttons in the right. I said sub because the right buttons named admin, cargomanager, transstaff, calc.(calculator) respectively are for depart between the forms and exit button for leaving the program and again showing start form.

Help part contains information about system. It has 3 sub parts. About Program, Contact Programmer, Technical Support. It has legal licence caution ,licence numbers in about program part. In contact programmer part it has the addresses of the

programmer company in detailed. Also the names and telephone numbers of the people who wrote the program. If user has a problem with system he/she can reach the numbers and addresses of programmers from the system too. Technical support part is for an introduction of program and contains some information about the usage of program.

Arrival cargo part contains the cargo informations which has arrived to branch. First of all sgstur company has totally 21 branches in the 5 main city. This information is kepted in the database branch table.

cityname	countyname	branchid	tel	address	faxno
istanbul	taksim	100	0 212 243 51 89	gümüşsuyu mh. osr	0 212 243 51 85
istanbul	bakırköy	111	0-212-572 22 95	zuhuratbaba mah.	0-212-572 25 09
istanbul	kadıköy	122	0-216-339 11 90	bağ sok. anadolu si	0-216-573 08 27
istanbul	kavacak	133	0-216-566 32 74	fahrettin kerim gök	0-216-347 93 78
istanbul	kartal	144	0-216-457 62 57	çavuşoğlu mah. erz	0-216-387 14 19
izmir	buca	155	0-232-452 66 96	forbest cad. 313 sc	0-232-452 66 43
izmir	hatay	166	0-232-277 87 25	ata cad. merkez me	0-232-277 87 66
izmir	bornova	177	0-232-461 79 43	2.sanayi sitesi 351	0-232-467 48 58
izmir	konak	188	0-232-375 65 53	doğanlar mah. 4.sa	0-232-375 65 53
izmir	çesme	199	0-232-712 87 71	16 eylül mah. 3008	0-232-452 66 43
izmir	karşıyaka	210	0-232-367 07 26	1675 sok. no:141-a	0-232-330 90 74
bursa	uludağ	221	0-224-757 03 46	selçuk mh. kılıçarsla	0-224-757 03 33
bursa	kirishane	232	0-224-713 00 48	osmaniye mh. vahit	0224-714 84 11
bursa	osmangazi	243	0-224-223 12 68	ahmetpaşa mh. fal	0-224-215 88 75
ankara	çankaya	254	0-312-231 32 72	uçarlı sok no:21/c y	0 312 426 15 79
ankara	batıkent	265	0-312-212 83 32	muammer aksoy ca	0-312-266 00 67
ankara	bağcıevler	276	0-312-354 20 43	kent koop.mah.cun	0-312-235 49 89
antalya	manavgat	287	0-242-746 66 71	aşağıhisar mah. dei	0-242-746 66 71
antalya	alanya	298	0242-517 29 05	zafer cad. no:5 av	0 242 512 74 14
antalya	kale	309	0242- 258 11 77	organize san böl. o	0 242 257 01 20
istanbul	avcılar	320	0-212-572 21 34	prof.turan güneş c	0 212 243 51 44

Figure 2.5 Branch Table

When a cargo send feom one branch to other system will record this and will write this to database of the destination branch. So that the when the cargo is reached to destination staff in the destination branch can approve this from by the system. Also when staff delivers the cargo to customer he will change the status of cargo as delivered from the system.

New Cargo part as the name implies for sending cargo from one branch to another. In this part customer comes to a branch and tells that he/she wants to send cargo to a city and the nearest branch in that city for customer's address. We want to remind that this company's working principle is that they only depart cargo from branch to branch. That means a there is no service from an adress to another departs is just between branches. In this part we record three main things for a cargo as branch information, customer information and cargo information. In branch information we record the sender and receiver branch's cities and counties names, in customer

information we record details about sender and receiver customer as name, surname, telephone, id number (citizenship no for sender), email addresses (for receiver that when cargo reached destination system automatically send mail to receiver that his/her cargo has arrived), address of sender and in cargo information part we record the date, serial no and invoice no of invoice, send type (cargo or dossier), send by (VIP or normal), pay type (cash or credit card), pay by (sender or receiver), weight of cargo, volume (if cargo selected) and the price. When the required information is filled staff will push the save button and will automatically save the records to database and will send info to destination branch's database. Also staff will push the print button and system will give a print document which will be an invoice for customer.

Branch Info part is just for information about branches. If staff wants to take an info about a branch he/she will just state the desired branch's city and county name than program automatically shows the branch id, phone, address, fax no of that branch.

Calculate Price part as named implies for calculating the price of a cargo. In this part staff will have to fill some required data. These are sender city name, receiver city name, send type (cargo dossier), send by (VIP, normal), weight, volume (if cargo selected). Then staff will push the calculate button and result is seen like in figure 2.4. VIP is a duty of sgur company that is customer chooses vip duty his/her cargo will arrive more quick but of course more expensive than others. Also there is an insurance choice if customer wants his/her cargo will include insurance with response of some price and the tax of the price is added to price and total amount will be calculated and appear on the screen. Finally the prices of cargo is determined by managers and assigned to the database tables.

Arrival Time part also an information part about a cargo's arriving time. For example staff will enter the required information as sender branch, receiver branch, send type (VIP, normal), send by (cargo, dossier) then press the calculate button and will see the arrival time and the km distances between two destinations. Arrival time is here changes depending on the send type and send by status. For example a customer wants VIP for his cargo so the arrival time will be less than normal send type. Also the arrival time is determined by the manager of cargo and assign to database tables.

Search Cargo part as name implies is for searching a cargo that is send from branch. This is done with invoice and serial number of invoice. Staff will enter these numbers and push search button, if record not found program will give a message ("no

record found”) else if record found staff can see all information of customer and also staff has permission of updating and deleting the selected record information.

We have also an employee department which is responsible with the information of workers and also it has a form in the program named as EmpMan where the information of workers are kept in to database tables.

Name	County	Town	Country	E-mail	Phone	Department	Address
Mesut	Gazi	Istanbul	Istanbul	sayesalone@yahoo.com	212 4533232	Transport	efiganugreemibgereng

Figure 2.6 EmpMan Form

EmpMan form as shown in figure 2.6 consist of four main parts and six sub parts. The sub parts as we told in other forms are for circulating between departments. That means the employee worker can access to trans staff form and can take a data, but if he/she wants to access to admin form the start page will appear and ask admin's username and password that also mean that he can access admin page with administrator's permission. Employee form has a right of circulating in cargostaff and transstaff forms only granting or revoking the other pages permissions is in admin's hand. We said we have 4 main parts in this form , these are;

- Find Employee
- Modify Employee
- New Employee
- Help

Help part as we explained before contains information about system. It has 3 sub parts. About Program, Contact Programmer, Technical Support. It has legal licence caution ,licence numbers in about program part. In contact programmer part it has the addresses of the programmer company in detailed. Also the names and telephone numbers of the people who wrote the porogram. If user has a problem with system he/she can reach the numbers and addresses of programmers from the system too. Technical support part is for an introduction of program and contains some information about the usage of program.

New Employee part as the name implies, when a new worker or employee starts working in company his/her personel information should be recorded so this part is for recording the employees information. This part contains and records name, surname, status (manager,staff,admin,asistant,driver,host), permission(if he/she uses program),department, personellId(citizenship number), county,town, country, email, phone, birthplace, bithdate and adress of employee. All the records are saved to the personel table in the database.

	personelid	name	surname	adres	email	birthdate	birthplace	permission	department	status	county	town	country	phone
	1000	Gökтуğ	Ataç	prof.turan güneş c.	goktugatac@yahoo	23.10.1984	Kırklareli	3	cargo	staff	zeytinburnu	istanbul	Turkey	0212 415 0140
	1001	Engin	Alan	askdijavofqwf	enginalan24@hotmail	14.02.1982	Tokat	2	transport	manager	okmeydanı	İstanbul	Turkey	0212 4325465
	1002	Mesut	Arık	sfigwugreejrnbgw	saysalane@yahoo	05.04.1984	Mardin	3	Transport	Staff	Gazi	İstanbul	İstanbul	0212 4533232
*														

Figure 2.7 Personel Table

Modify employee part is related with new user part because when user records a new employee to database he/she can see every person that she had recorded before. If there is a change in a workers information or a mistake will be smoothed first user selects the desired work and see all information in the textboxes than can change the desired info. Also when a worker leaved from company at that moment user can delete the leaving personel from the system and database. Here there is a point that user must be careful personelId (citizenship number), is a unique number that means it is primary key for that table for this reason there is no one that can have same personel Id also user cannot modify this info.

Find employee part is also related with new user and modify user parts because if user wants to see an employees informations he has to know his personellId or press the advanced searc buton and will fill personel's name or surname. User willl enter personelId of desired worker then press search button, if there is a match in database the records information will appear in the screen if not a message will appear and say

“no match found”. Also he can enter personel’s name or surname or both then find personel an see the information. That is all for EmpMan form.

Now let us look to a new form CargoMan which is the manager of cargo department. As we described in the cargostaff department cargo manager controls and coordinates the cargo departments main work.

Figure 2.8 CargoMan Form

CargoMan form as seen in the figure 2.8 contains six main parts and three sub parts. The sub parts are cargostaff,admin,calc.(calculator) parts which is for circulating between departments. Also here if cargomanager wants to access to admin part system will ask username and password for this.

As we said we have 6 main parts in CargoMan form;

- Branch Info
- Employee Info
- Price Info
- Arrival Time Info
- Account
- Help

Help part as we explained before, contains information about system. It has 3 sub parts. About Program, Contact Programmer, Technical Support. It has legal licence caution ,licence numbers in about program part. In contact programmer part it has the addresses of the programmer company in detailed. Also the names and telephone

numbers of the people who wrote the program. If user has a problem with system he/she can reach the numbers and addresses of programmers from the system too. Technical support part is for an introduction of program and contains some information about the usage of program.

Account part is an information part that the manager can take and see the daily, monthly or yearly accounts in the company of which the cargo department done. So he can analysis the the account information from this part.

Arrival Time Info part as name implies about distances and the arrival times that a cargo departures and arrives. If we look to figure 2.8 the shape of page is like view. In the upper side manager can see the distance and arrival time when he/she enters required information (destinations, sendby, sendtype) and press calculate button then desired data will be seen. As far as manager is designating the times and distances for cargo arrival times so in the lower part he can update or reenter the new arrival time for cargo or dossier. Here as we see there is a map of Turkey and a link below of it, this link ties the user to an internet page of (<http://www.kgm.gov.tr>) to see the distances between railways in turkey for calculating approximate arrival times. All these are kept to arrivaltime table in database.

	autonumber	fromb	tob	km	sendby	sendtype	arrivalt
▶	1	istanbul	izmir	565	cargo	normal	24
	2	istanbul	izmir	565	dossier	VIP	18
	3	istanbul	izmir	565	cargo	VIP	24
	4	istanbul	izmir	565	dossier	normal	30
	5	istanbul	ankara	453	cargo	normal	24
	6	istanbul	ankara	453	dossier	VIP	18
	7	istanbul	ankara	453	cargo	VIP	24
	8	istanbul	ankara	453	dossier	normal	30
	9	istanbul	bursa	243	cargo	normal	36
	10	istanbul	bursa	243	dossier	VIP	18
	11	istanbul	bursa	243	cargo	VIP	24
	12	istanbul	bursa	243	dossier	normal	18
	13	istanbul	antalya	724	cargo	normal	36
	14	istanbul	antalya	724	dossier	VIP	18
	15	istanbul	antalya	724	cargo	VIP	24
	16	istanbul	antalya	724	dossier	normal	30

Figure 2.9 Arrivaltime Table

In this table we have 100 records but this figure just shows a part of it. Here is only for İstanbul to other cities. If you pay attention arrivaltime is various depending on send by and send type adn km between two cities.

Price info part 's working principle is like arrival time info part. Actually the manager first controls or views the price of cargo or dossier between two cities depending on kilogram of cargo then he can analysis the price and go to lower part and update or reenter the new value price for this cargo. When designating the new value for

cargo manager needs distances between 2 destinations so he can again use the link under the map. Also here all the variations or adding or deleting processes is held on 2 database tables. First one is for dossier price other is for cargo price.

Again here we have nearly one hundred of records but figure just shows the istanbul records. Also here the price various depending on kilogram, distance between cities. And as we see for a dossier maximum 10 kg is acceptable

autonum	fromc	toc	kg	price
1000	istanbul	izmir	0-2	6
1001	istanbul	izmir	2-6	15
1002	istanbul	izmir	6-10	24
1003	istanbul	antalya	0-2	8
1004	istanbul	antalya	2-6	21
1005	istanbul	antalya	6-10	30
1006	istanbul	bursa	0-2	6
1007	istanbul	bursa	2-6	15
1008	istanbul	bursa	6-10	24
1009	istanbul	istanbul	0-2	5
1010	istanbul	istanbul	2-6	13
1011	istanbul	istanbul	6-10	22
1012	istanbul	ankara	0-2	6
1013	istanbul	ankara	2-6	15
1014	istanbul	ankara	6-10	24

Figure 2.10 Dossierprice Table

Also we have a cargo price table like dossier price. Here maximum 100 kilogram is acceptable for cargo. And the price is determined according to kilogram and distances between cities. But here volume of cargo is also important we will cover that part in the next chapter while explaining code parts.

autonumber	fromc	toc	kg	price
10	istanbul	izmir	0-10	7
11	istanbul	izmir	10-30	17
12	istanbul	izmir	30-60	30
13	istanbul	izmir	60-100	42
14	istanbul	antalya	0-10	11
15	istanbul	antalya	10-30	23
16	istanbul	antalya	30-60	38
17	istanbul	antalya	60-100	51
18	istanbul	ankara	0-10	7
19	istanbul	ankara	10-30	17
20	istanbul	ankara	30-60	30
21	istanbul	ankara	60-100	42
22	istanbul	bursa	0-10	7
23	istanbul	bursa	10-30	17
24	istanbul	bursa	30-60	30
25	istanbul	bursa	60-100	42
26	istanbul	istanbul	0-10	4
27	istanbul	istanbul	10-30	11
28	istanbul	istanbul	30-60	22
29	istanbul	istanbul	60-100	34

Figure 2.11 Cargo Price

Employee Info part is for information of employee for the general of company. Manager can directly find and see an employee's information with entering search criterias of employee. Manager can update or delete an employee but if you pay attention as this is a cargo manager for that reason he can only update or delete who works in cargo department that means he cannot update or delete in other departments personels if he tries to do this system will aoutmtically warn user that "you can only update cargo department personal".

Branch Info part as we described before is just for information about branches. If user wants to take an info about a branch he/she will just state the desired branch's

city and county name than program automatically shows the branchid, phone, adress, fax no of that branch. That is all for CargoMan form.

Now let us consider our second main department's staff part. For the analysis of transport department we had a lot of investigations from the real life companies. Finally we found some main things and constructed our form.

Figure 2.12 TransStaff Form

TransStaff form as seen in the figure 2.12 contains five main parts and four sub parts. The sub parts are cargo staff , admin, Tran manager , calc.(calculator) parts which is for circulating between departments. Also here if user wants to access to admin part system will ask username and password for this.

As we said we have 5 main parts in this form these are;

- Ticket
- Reservation Change
- Sale Change
- Branch Info
- Help

Ticket part is the main framework of transport department because every transact is supplied from here. For example a customer comes and wants to go to a place and want to reserve or sale a seat, after stating the exact time and place staff will look for available seats in that bus and make the reservation. If you pay your attention when staff chooses the expedition the status of seats for that bus is appears so that customer

can see where is available to seat. Also when staff press to filled seats on the bus seat (figure 2.12) he/she can see the information of customer in the lower part as name, surname.

Reservation change part as name implies when a change or cancel happens in the reservation staff can change it. For example a customer arrives and states that he had a reservation for 12.00 bus from Istanbul to İzmir but he states that he cannot depart at that time so he has to change his reservation to next expedition at 16.00 so that moment staff will open the reservation change part and do this. Also the same thing for canceling a reservation is valid. System also warns the user to warn the customer to be at the station 30 minutes before the expedition.

Sale change part's work principle is same as reservation change part. Here again if there is wrong in ticket or customer wants to refund his/her ticket staff can refund money and cancel the sale and reservation from the expedition but again here there is a exception customer can refund his/her ticket till maximum one hour before expedition.

Branch Info part is just for information about branches. If staff wants to take an info about a branch he/she will just state the desired branch's city and county name than program automatically shows the branchid, phone, adress, fax no of that branch.

Help part as we explained before, contains information about system. It has 3 sub parts. About Program, Contact Programmer, Technical Support. It has legal licence caution, licence numbers in about program part. In contact programmer part it has the addresses of the programmer company in detailed. Also the names and telephone numbers of the people who wrote the program. If user has a problem with system he/she can reach the numbers and addresses of programmers from the system too. Technical support part is for an introduction of program and contains some information about the usage of program.

Actually for transport department we have expedition, bus, seat tables in the database. Expedition table is for creating new voyages, seat table is for recording a unique expedition number and seat numbers, in the bus table we record our busses in our fleet.

After Tran staff we have also a Tran manager named as Trasman form. Like in cargo department the main responsibility of manager is to direct transport department and coordinate the expeditions.

Figure 2.13 TransportMan

TransMan form as seen in the figure 2.13 contains seven main parts and four sub parts. The sub parts are cargostaff, admin, calc.(calculator) parts which is for circulating between departments. Also here if user wants to access to admin part system will ask username and password for this.

As we said we have 7 main parts in this form these are;

- Find expedition
- Create expedition
- Modify Expedition
- Busses
- Personnel
- Account
- Help

Find expedition part is for searching the expeditions that the manager was created. First of all manager can directly see all the expeditions also he can search it advanced by entering date and destinations. The main benefit of this part is to reach the expedition information for creating new expedition.

Create expedition part as the name implies is the main responsibility of manager as creating expeditions daily. This part is very important because manager has to analysis and coordinates busses, drivers, hosts and construct expeditions every end of the day for the following day. While he is creating new expedition he has to fill date,

new expedition number , destinations, price, driver name, host name, bus Id, bus plate and when he press the create button automatically expedition is created and saved to database expedition table. Also after creating, cargostaff can see this expedition and can reserve or sale tickets for that expedition.

from a	destination	expedition no	dates	times1	times2	price	drivername	driversurname	hostname	hostsurname	busnumber	busplate
izmir	bursa	1000	30.06.2006	10	45	35	mesut	ank	süleyman	kerime	123	34UN4032
bursa	antalya	1001	31.06.2006	11	15	45	mehmet	dak	ahmad	barrabi	234	34UB3452
*												

Figure 2.14 Expedition Table

Modify expedition part as again name implies for modifying or updating an expedition that the manager was created. For example he created an expedition and he realized that he had a mistake with hour or has to change the driver or host so for all of these he will use this part. First he finds the expedition that he created before then he chooses the desired expedition and can reach to data that will be modified.

Busses part is considered when this company buys new bus or new cargo vehicle they will record this to database. The logic of recording new vehicles to system is for using them in create expedition part as while choosing the vehicle for expedition.

Personnel part is for information part that the manager can access to information of transport department workers. When he opens this part all the transport employees appears and then the selected personnel's information will be shown.

Account part is information part too that the manager can take and see the daily, monthly or yearly accounts in the company of which the transport department done. So he can analysis the account information from this part. The main idea of this part to see the balances of transport department as income and expenditure and with reports he can make analysis.

Help part as we explained before, contains information about system. It has 3 sub parts. About Program, Contact Programmer, Technical Support. It has legal licence caution, licence numbers in about program part. In contact programmer part it has the addresses of the programmer company in detailed. Also the names and telephone numbers of the people who wrote the program. If user has a problem with system he/she can reach the numbers and addresses of programmers from the system too. Technical support part is for an introduction of program and contains some information about the usage of program.

So we have only two forms left. Account man and Admin forms. If we start with account form we can say that this department is a general department for most of companies and maybe the most important department for a company.

	03-05-2003	05-05-2004
INCOME		YTL
EXPENDITURE		YTL
BALANCE		YTL

Figure 2.15 Account Man Form

Account Man form as seen in the figure 2.15 contains six main parts and four sub parts. The sub parts are cargostaff, Tran staff, admin, calc.(calculator) parts which is for circulating between departments. Also here if user wants to access to admin part system will ask username and password for this.

As we considered that we have 6 main parts in this form these are;

- Transport Income
- Transport Expenditure
- Cargo Income
- Cargo Expenditure
- Salary
- Help

Transport Income part is for reporting the incomes of transport department. This is done or let's say the total incomes are calculated with Transstaff's sales. When Trans staff sales a ticket the transact is automatically comes to account's database so that user can see directly all sales items.

Transport Expenditure part as the name implies is about expenditure that transport department have. Here the domain expenditure is calculated from expedition.

User first enters the expedition number, then will assign the expenditure belongs to that expedition. Taccount table in database stores these records. The contents of table is, expedition no, expenditure, dates, times, driver name, driver surname, host name, host surname, bus number, bus plate, destination. That's means user can also take report according to expedition no or date or time etc.

Also the important thing is all the expenditures for a expedition (fuel, foodstuffs, taxes etc) are combined and calculated then entered as one expenditure.

Cargo Income part is again for reporting the incomes of cargo department. This is also done or let's says the total incomes are calculated with Cargo staff's sales. When Cargo staff send a cargo the transact is automatically comes to account's database so that user can see directly all sales items. All incomes are taken from departurecargo table where all information is kept for a cargo.

Cargo Expenditure part as again the name implies is about expenditure that cargo department have. Here the domain expenditure is calculated from daily or let's say in the end of each date. User first enters the date, then will assign the expenditure belongs to that day. Caccount table in database stores these records. The contents of table is, date and expenditure. Here again all expenditure (fuel, foodstuffs, taxes etc) are combined and calculated then entered as one expenditure. That's means user can also take report according to date.

Salary part is for the record of employees salaries in the company. Here user records every worker's transact in the beginning of month. Also he can change the amount of salary of an employee. Also he can make calculation with total amount of transacts. He can see all employees and calculate total delivered money to worker.

Help part as we explained before, contains information about system. It has 3 sub parts. About Program, Contact Programmer, Technical Support. It has legal licence caution, licence numbers in about program part. In contact programmer part it has the addresses of the programmer company in detailed. Also the names and telephone numbers of the people who wrote the program. If user has a problem with system he/she can reach the numbers and addresses of programmers from the system too. Technical support part is for an introduction of program and contains some information about the usage of program.

Finally, the framework of the system or let's say the coordinator of system is the administrator. The form of administrator is named as admin.

Figure 2.16 Admin Form

Admin Man form as seen in the figure 2.16 contains six main parts and seven sub parts. As this is admin form he/she can access all other departments and coordinate the system. But for security when admin accesses to another form and wants to return back system will ask username and password. So we told we have 6 main parts in administration form.

- Add User
- Edit User
- Database
- Permission
- Back Up
- Help

Add user part, if we remember in the beginning of this chapter we started with start form. If user wants to access to system he/she has to type username and password. The problem is to have a username and a password. In this part admin gives username and password to the user of system. If you pay attention in username is unique. That means all users in system has different usernames but they can have same password. For that reason admin first of all controls the username before giving it to user. He types the username that he will give for new system user then the program automatically controls it in the database. If there is no username as he typed then system gives message that he

can give. If not again program warns user that this is not a unique, there is already a username like you typed. After this admin assigns username, password, department, permission, personelid, name and surname.

Edit User part is for framing, updating or deleting users information. In this part admin first finds the system user that he/she will use than modify it. All the modifications and views are supplied from password table in database.

	username	idnumber	permission	department	name	surname	personelid
▶	ad	111	1	admin	sinan	öz	1232333244
	ad1	1987	2	employee	ali	ali	5678
	as	888	2	account	Mesut	arik	7777773327
	cm	222	2	cargo	ozan	akkoca	6666345666
	cs	444	3	cargo	göktuğ	ataç	5553465555
	em	666	2	employee	engin	alan	4423423444
	tm	333	2	transport	ismail	hakkı	22132578222
	ts	555	3	transport	sinan	özerenler	1134555411
*							

Figure 2.17 Password Table

Database part is for viewing the all the records in the database. Admin selects the table that he wants to see then he can view all the data. As we have 16 tables and he can access which ever he wants. But here he can not modify any table if he wants he can access other departments and can modify easily.

Name	Owner	Type	Create Date
arrivalcargo	dbo	User	11.05.2006 14:48:06
arrivaltime	dbo	User	12.04.2006 00:57:02
branch	dbo	User	10.04.2006 23:06:36
bus	dbo	User	05.05.2006 15:54:47
caccount	dbo	User	18.05.2006 15:39:26
cargoprice	dbo	User	15.04.2006 02:29:38
departurecargo	dbo	User	19.05.2006 14:39:23
dossierprice	dbo	User	15.04.2006 18:31:38
expedition	dbo	User	05.05.2006 16:02:30
password	dbo	User	13.05.2006 16:53:48
personel	dbo	User	03.05.2006 11:08:27
reservation	dbo	User	05.05.2006 15:58:26
salary	dbo	User	18.05.2006 16:47:27
seat	dbo	User	05.05.2006 16:00:27
taccount	dbo	User	18.05.2006 15:38:17
ticketsale	dbo	User	05.05.2006 15:58:46

Figure 2.18 All Database Tables

As we see our program uses 16 tables. Also in this part while admin selects the database he can choose table that is related with department because in the program databases are grouped according to departments.

Permission part as the name implies used for granting or revoking permissions to the user. For example admin wants to grant the update statement in a table for a user. First he selects the table and sees the permissions on the table. Then he can change the permission of that table for the users.

Back Up part is almost the most important for a program that records and uses database. Because losing data means losing money for a company. As this is very risky this part's responsibility is to take back up and save to another place. Admin has to make this operation at the end of each day as from program or from inside the operating system.

Help part as we explained before, contains information about system. It has 3 sub parts. About Program, Contact Programmer, Technical Support. It has legal licence caution, licence numbers in about program part. In contact programmer part it has the addresses of the programmer company in detailed. Also the names and telephone numbers of the people who wrote the program. If user has a problem with system he/she can reach the numbers and addresses of programmers from the system too. Technical support part is for an introduction of program and contains some information about the usage of program.

CHAPTER 3

EXAMINATION OF CODE STRUCUTRE

In chapter two we always stated that no codes example is given all codes will be assign in the chapter three and we always say while program do something we used word "automatically". In this chapter we will see how the program do all these automatically and which codes does it uses for all these.

We have start form as we described in chapter two. We said the working principle of that page is very easy and just with if-else statements it chooses the right form for the user and accesses to form. The code for application is ;

```
string sqlQuery = "SELECT permission,department,name,surname FROM  
password where username = '\"+textBox1.Text.ToString()+" \' and idnumber =  
\"'+textBox2.Text.ToString()+"\"";
```

```
sqlDataAdapter1.SelectCommand.CommandText = sqlQuery;
```

```
dataSet1.Clear();
```

```
int numberOfRowsFeched = sqlDataAdapter1.Fill(dataSet1,"password");
```

```
if (numberOfRowsFeched > 0)
```

```
{
```

```
    dt=dataSet1.Tables["password"];
```

```
    dataTable1=dataSet1.Tables["password"];
```

```
    label4.Text=dataTable1.Rows[0]["name"].ToString();
```

```
    label5.Text=dataTable1.Rows[0]["surname"].ToString();
```

```
    if((dt.Rows[0][0].ToString()=="3") && (dt.Rows[0][1].ToString()=="transport"))
```

```
    {
```

```
        transtaff frm=new transtaff();
```

```
        frm.strname1=label4.Text;
```

```
        frm.strsurname1=label5.Text;
```

```
        frm.Show();
```

```
        this.Hide();
```

```
    }
```

```
else if((dt.Rows[0][0].ToString()=="3") && (dt.Rows[0][1].ToString()=="cargo"))
```

```
{
```

```
    cargostaff frm1=new cargostaff();
```

```
    frm1.strname=label4.Text;
```

```
    frm1.strsurname=label5.Text;
```

```
    frm1.Show();
```

```
    this.Hide();
```

```
}
```

```

else if((dt.Rows[0][0].ToString()=="2") && (dt.Rows[0][1].ToString()=="employee"))
{
    employee1 frm3=new employee1();
    frm3.Show();
    this.Hide();
}
else if((dt.Rows[0][0].ToString()=="2") && (dt.Rows[0][1].ToString()=="cargo"))
{
    cargoman frm4=new cargoman();
    frm4.Show();
    this.Hide();
}
else if((dt.Rows[0][0].ToString()=="2") && (dt.Rows[0][1].ToString()=="transport"))
{
    transportman1 frm5=new transportman1();
    frm5.Show();
    this.Hide();
}
else if((dt.Rows[0][0].ToString()=="2") && (dt.Rows[0][1].ToString()=="account"))
{
    accountman frm6=new accountman();
    frm6.Show();
    this.Hide();
}
else
{
    admin frm7=new admin();
    frm7.Show();
    this.Hide();
}
}
else
{
    MessageBox.Show("Please try again.");
    textBox1.Text="";
    textBox2.Text="";
    textBox1.Focus();
}
}

```

As we see the code is very easy to mention. First part is for ,when the user types his/her user username and password system goes to database and queries the password table. If there is record like user typed than it takes the permission and department then using if –else statement it chooses the right form to open. For example a user enters his username and password and system finds that his permission is “3” and his department

is “cargo” so system checks the statements and finds that he is a cargo staff and so opens the cargo staff form for that user.

Actually, we have some main codes for relation between database , or for windows controls. Now we are going to analysis mainly these codes.

3.1 Selecting Data From Database

```
String sqlQuery="Select cityname,countyname,branchid,tel,address,faxno from branch
where cityname=  \"'+comboBox2.Text.ToString()+'\"  \'  and countyname=
\"'+comboBox3.Text.ToString()+'\""; // sql query part
branchDataAdapter.SelectCommand.CommandText=sqlQuery;
branchSet.Clear();// clear the contents of dataset
Int fec1=branchDataAdapter.Fill(branchSet,"branch");// if there is a row like query
if(fec1>0){dataTable1=branchSet.Tables["branch"];
brcity.Text=dataTable1.Rows[0][0].ToString();
brcounty.Text=dataTable1.Rows[0][1].ToString();
brid.Text=dataTable1.Rows[0][2].ToString();
brphone.Text=dataTable1.Rows[0][3].ToString();
braddress.Text=dataTable1.Rows[0][4].ToString();
brfax.Text=dataTable1.Rows[0][5].ToString();
}
```

First of all we write the sql query which we state the table and desired rows. Then using data adapters and data Sets we take the information from the database and view it on the forms.

3.2 Inserting Data to Database

3.2.1 Using Stored Procedures

```
CREATE PROCEDURE [Insert exp]
@fr nvarchar(15),
@des nvarchar(15),
@dt nvarchar(15),
@tm nvarchar(10),
@pr nvarchar(10),
@dr nvarchar(30),
@ho nvarchar(30),
@bn nvarchar(10),
```

```
@bp nvarchar(10)
```

```
AS
```

```
InsertInto
```

```
expedition(from_a,destination,dates,times1,times2,price,driver,host,busnumber,busplate
```

```
) values (@fr,@des,@dt,@tm,@pr,@dr,@ho,@bn,@bp)
```

```
GO
```

Then we write insert codes to the program ;

```
sqlConnection1.Open();
```

```
SqlCommand cmdInsert =new SqlCommand("Insertobje2",this.sqlConnection1);
```

```
cmdInsert.CommandType=CommandType.StoredProcedure;
```

```
SqlParameterparamfirstname=cmdInsert.Parameters.Add
```

```
("@fn",System.Data.SqlDbType.NVarChar,15);
```

```
SqlParameterparamsurname=cmdInsert.Parameters.Add
```

```
("@s",System.Data.SqlDbType.NVarChar,15);
```

```
paramfirstname.Value=txtname1.Text;
```

```
paramsurname.Value=txtsurname1.Text;
```

```
cmdInsert.ExecuteNonQuery();
```

```
sqlConnection1.Close();
```

3.2.2 Using SQL Query

We write to following codes.

```
sqlDataAdapter1.Fill(dataSet1_1,"expedition");
```

```
dataTable1_1 = dataSet1_1.Tables["expedition"];
```

```
newRow = dataTable1_1.NewRow();
```

```
newRow["expedition_no"] = lbexpnum.Text;
```

```
newRow["from_a"] = cmbfrom1.Text;
```

```
newRow["destination"] = cmbdestination1.Text;
```

```
newRow["dates"] = dtpdate1.Text;
```

```
dataTable1_1.Rows.Add(newRow);
```

```
sqlDataAdapter1.Update(dataSet1_1,"expedition");
```

```
Application.DoEvents();
```

```
dataSet1_1.AcceptChanges();
```


3.3 Updating Data in Database

```
sqlDataAdapter1.Fill(dataSet1,"personel");
dataTable1=dataset1.Tables["personel"];
targetRow = dataTable1.Rows[lstpersonel.SelectedIndex];
targetRow.BeginEdit();
targetRow["name"]=txtname.Text;
targetRow["surname"]=txtsurname.Text;
targetRow["status"]=cmstatus1.Text;
targetRow["permission"]=txtpermission.Text;
targetRow["birthplace"]=txtbirthplace.Text;
targetRow["birthdate"]=txtbirthdate.Text;
targetRow["adres"]=txtaddress.Text;
targetRow["county"]=txtcounty.Text;
targetRow["town"]=txttown.Text;
targetRow["country"]=txtcountry.Text;
targetRow.EndEdit();
sqlDataAdapter1.Update(dataSet1,"personel");
Application.DoEvents();
dataset1.AcceptChanges();
MessageBox.Show("Update Successful");
dataset1.Clear();
```

First of all here dataset1 send the data to dataadapter that it catches in personel table. Then change the info with assigned values. At the end to be the changes valid we should say dataset AcceptChanges ().

3.4 Delete Data From Database

```
string sqlq="DELETE from expedition where expedition_no=
\""+lbexpnum2.Text.ToString()+" \";
sqlDataAdapter1.SelectCommand.CommandText=sqlq;
targetRow = dataTable1.Rows[lstexpmod.SelectedIndex];
targetRow.Delete();
sqlDataAdapter1.Update(dataSet1,"expedition");
Application.DoEvents( );
dataset1.AcceptChanges();
```

```

MessageBox.Show("Record deleted.");
dataSet1.Clear();

```

First of all we find the desired row that we want to delete by sql query. Then assigning a simple delete command we can automatically delete a row from database.

3.5 Windows Control

3.5.1 Adding Item to ComboBox

```

comboBox2.Items.Add("İstanbul");
comboBox2.Items.Add("İzmir");
comboBox2.Items.Add("Ankara");
comboBox2.Items.Add("Bursa");
comboBox2.Items.Add("Antalya");

```

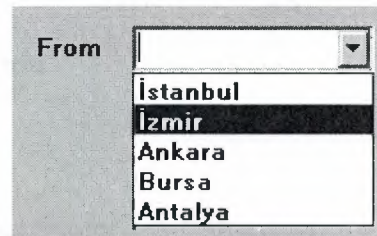


Figure 3.1 ComboBox1

```

for(i=1;32>i;i++)
{
    if((0<i) && (i<10))
    {
        cmday1.Items.Add("0"+i);
    }
    else
    {
        cmday1.Items.Add(i);
    }
}

```

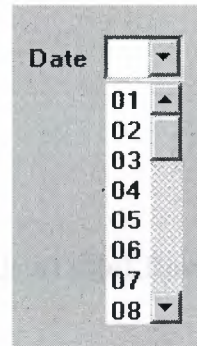


Figure 3.2 Combobox2

3.5.2 Filling DataGrid

```

string sql1="Select dates,times,expedition_no,ticketno,price from ticketsale where
dates='"+lbhid1.Text.ToString()+"'";
sqlDataAdapter6.SelectCommand.CommandText=sql1;
dataSet6_1.Clear();
sqlDataAdapter6.Fill(dataSet6_1,"ticketsale");
this.dataGrid1.DataSource=dataSet6_1;
this.dataGrid1.DataMember=dataSet6_1.Tables[0].TableName;

```


	dates	times	expedition_	ticketno	price
▶	11.05.2006	01:15	1000	360	45
	11.05.2006	01:15	1000	380	45
	11.05.2006	01:15	1000	420	45
	11.05.2006	01:15	1000	440	45
	11.05.2006	01:15	1000	500	45
	11.05.2006	01:15	1000	520	45
	11.05.2006	01:15	1000	720	45

Figure 3.3 DataGrid

3.5.3 Filling ListBox

```

string sqlq="SELECT * from expedition";
sqlDataAdapter1.SelectCommand.CommandText=sqlq;
dataSet1.Clear();
int fec=sqlDataAdapter1.Fill(dataSet1,"expedition");
if(fec>0)
{
    dataTable1 = dataSet1.Tables[0];
    lstexpmod.Items.Clear( );
    foreach (System.Data.DataRow dataRow in dataTable1.Rows)
    {
        lstexpmod.Items.Add(" "+
        dataRow["times1"]+": "+
        dataRow["times2"]+" "+
        dataRow["dates"] + " "+
        dataRow["busplate"]+" "+
        dataRow["from_a"]+" "+
        dataRow["destination"]);
    }
}

```

01:15	11.05.2006	35AR 432	İstanbul	İzmir
01:00	12.05.2006	34GR 234	İstanbul	İzmir
00:00	12.05.2006	34GR 234	İstanbul	İzmir
02:30	12.05.2006	34GR 234	İstanbul	İzmir
02:15	16.05.2006	34GR 234	İstanbul	İzmir
02:30	18.05.2006	34GR 234	İstanbul	İzmir
01:15	28.05.2006	34GR 234	İzmir	Ankara

Figure 3.4 ListBox

3.5.4 Creating DialogBox

```
DialogResult dlgRes=MessageBox.Show  
("Are you sure to delete record of bus?  
","Delete Bus",  
MessageBoxButtons.YesNo,  
MessageBoxIcon.Question);
```

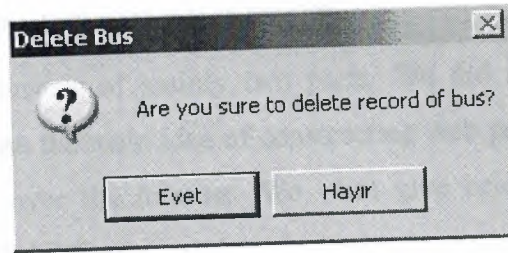


Figure 3.5 DialogBox

CHAPTER 4

WEB APPLICATION PART OF PROJECT

The web page part of our project consists of mainly two parts. We did not constitute too many parts for web page because the main idea of constructing web page is for information and accessing the work over the internet. We want give briefly information before start about HTML and ASP.NET;

4.1 HTML and XHTML

The main purpose of HTML is to enable web authors to specify structural information about their pages. HTML and XHTML are document-layout and hyperlink-specification languages. They define the syntax and placement of special, embedded directions that aren't displayed by the browser but tell it how to display the contents of the document, including text, images, and other support media. The languages also tell you how to make a document interactive through special hypertext links, which connect your document with other documents - on either your computer or someone else's - as well as with other Internet resources.

HTML and XHTML documents consist of text, which defines the content of the document, and tags, which define the structure and appearance of the document. The structure of an HTML document is simple, consisting of an outer `<html>` tag enclosing the document head and body.

Each document has a head and a body, delimited by the `<head>` and `<body>` tags. The head is where you give your document a title and where you indicate other parameters the browser may use when displaying the document. The body is where you put the actual contents of the document. This includes the text for display and document-control markers (tags) that advise the browser how to display the text. Tags also reference special-effects files, including graphics and sound, and indicate the hot spots (hyperlinks and anchors) that link your document to other documents.

For the most part, tags - the markup elements of HTML and XHTML - are simple to understand and use, since they are made up of common words, abbreviations, and notations. For instance, the `<i>` and `</i>` tags respectively tell the browser to start and stop italicizing the text characters that come between them. Accordingly, the syllable "simp" in our barebones example above would appear italicized on a browser display.

The HTML and XHTML standards and their various extensions define how and where you place tags within a document. Let's take a closer look at that syntactic sugar that holds together all documents.

Every tag consists of a tag name, sometimes followed by an optional list of tag attributes, all placed between opening and closing brackets (< and >). The simplest tag is nothing more than a name appropriately enclosed in brackets, such as <head> and <i>. More complicated tags contain one or more attributes, which specify or modify the behavior of the tag.

According to the HTML standard, tag and attribute names are not case-sensitive. There's no difference in effect between <head>, <Head>, <HEAD>, or even <HeaD>; they are all equivalent. With XHTML, case is important: all current standard tag and attribute names are in lowercase. For both HTML and XHTML, the values that you assign to a particular attribute may be case-sensitive, depending on your browser and server. In particular, file location and name references - or uniform resource locators (URLs) - are case-sensitive.

Tag attributes, if any, belong after the tag name, each separated by one or more tab, space, or return characters. Their order of appearance is not important. A tag attribute's value, if any, follows an equals sign (=) after the attribute name. You may include spaces around the equals sign, so that width=6, width = 6, width =6, and width= 6 all mean the same. For readability, however, we prefer not to include spaces. That way, it's easier to pick out an attribute/value pair from a crowd of pairs in a lengthy tag.

With HTML, if an attribute's value is a single word or number (no spaces), you may simply add it after the equals sign. All other values should be enclosed in single or double quotation marks, especially those values that contain several words separated by spaces. With XHTML, all attribute values must be enclosed in double quotes. The length of the value is limited to 1,024 characters.

Comments are another type of textual content that appears in the source HTML document but is not rendered by the user's browser. Comments fall between the special <!-- and --> markup elements. Browsers ignore the text between the comment character sequences.

There must be a space after the initial <!-- and preceding the final -->, but otherwise you can put nearly anything inside the comment. The biggest exception to this rule is that the HTML standard doesn't let you nest comments.

4.2 OVERVIEW OF ASP.NET

ASP.NET is Microsoft's dynamic website technology, enabling developers to create data-driven websites using the .NET platform. ASP.NET is an object-oriented, event-driven development platform for writing Web-based applications. Before .NET, Active Server Pages was the core Microsoft technology for developing applications that ran through the browser. ASP was a great platform, and it truly revolutionized the way Web applications were written, but it had lots of room for improvement. With ASP.NET, the gap between writing Windows-based applications and Web-based applications has been closed.

Because ASP.NET is based on the .NET Framework, the same classes in the Framework class library (FCL) are available to all .NET-based applications. That means the same coding model that you use to write Windows Forms applications is used to write ASP.NET applications. It also means you can write ASP.NET applications in any .NET language.

One of the drawbacks to writing ASP applications is the scalability issue. Because ASP pages are written in script, the code must be interpreted each time a page is accessed. To improve performance, developers write complex caching schemes, use different session state handling optimizations to improve page throughput, and move code into compiled COM components to increase performance—that all changes with .NET. All ASP.NET applications are compiled. There's no interpreted script of server-side code, and the ASP.NET runtime is a multithreaded asynchronous application, so the core infrastructure is more scalable than ASP.

It's much easier to build a rich user interface into Web application in ASP.NET 2.0 than it was in previous versions. Master Pages let building pages based on existing templates of markup and code. ASP.NET 2.0 wizards make implementing navigation easier. In addition, adding personalization to ASP.NET 2.0 pages thanks to features like the user profile. This month I'll take a quick tour through another feature that makes implementing ASP.NET applications easier, themes which let to skin control pages and achieve visual consistency with minimal effort.

The beauty of ASP.NET 2.0 themes is that, much like Windows® XP themes, they allow to radically change the appearance of a set of pages by applying minimal changes. In a themed control, visual properties can be set and changed in a single shot.

With ASP.NET themes, you can build skinned controls as long as the control supports templates.

A theme is an attribute of the page class that you can set in various ways. Once the theme is set, the page ensures that all of its controls will render according to the visual settings defined in the theme. Themes let to keep style markup out of ASPX code so you can write pages without concerning yourself with the style of controls and the graphics around them. Later, designers can create the theme.

In ASP.NET 2.0, a theme is made up of various files all installed in a given folder. The name of the folder determines the name of the theme. In the folder, there are a few types of files: a skin file, a CSS file, and optionally a subfolder of images or other auxiliary files such as XSLT and text files. Figure 4.1 shows an example.

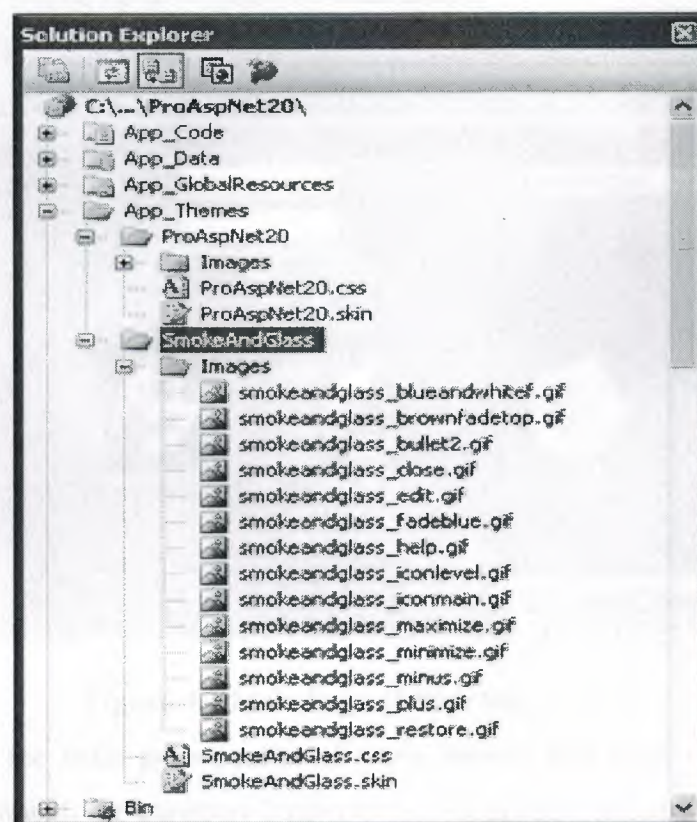


Figure 4.1 Custom Themes in ASP.NET 2.0

To be recognized by an application, a theme folder must either be located under the `App_Themes` folder below the application's root, or in a global folder. A theme located under the `App_Themes` folder at the root of the application is said to hold local themes as only the specific application can use it. Global themes are contained in child directories located under the following path:

`%WINDOWS%\Microsoft.Net\Framework\[version]\Asp.NetClientFiles\Themes`

4.3 Web Project Overview

We described that we had 2 main parts in our web application part. The cargo and transportation part. The main idea of transport department is to supply customers to buy tickets over the web site. Also the main idea of cargo department is to supply customers to follow their cargo if it is delivered or arrived by using their invoice numbers. Also we have a main page which is like ;



Figure 4.2 Main Page of Web Site

As you see main page consist of some buttons and short cuts for cargo and transport department. Sg transport button is for accessing transport part like sg cargo button is for cargo part. Customers can also make reservation over phone by calling call center which works 24 hours. Also site have a member list if customer gets an account and sign with his/her card number and password his/her own page will be displayed. In own page there will be messages, his previous purchases or cargo orders records. Also customers can see the branch information from site as address, telephone number or fax number of any branch.

In the transport page as we said customers can see the expeditions for desired day and can make reservation over the site.



Figure 4.3 Transport Page

Expedition part shows all expeditions for desired day and desired destinations. Customer first selects the destinations then the date of departure finally expeditions seen on the screen. From that expedition's customer can make reservation for desired expedition. Also for information the weather conditions and road status can be informed by guiding to related web sites.

In about us part the mission of the company and the history of company is mentioned. Communication part is for information of branches. Baggage part consists of agreements of cargo and transportation rules. It also consists of insurance agreements for cargo.

In the trans staff page the aim and mission of department is summarized as to create the atmosphere for a safe and comfortable trip for our passenger, to renew ourselves all the time and to serve our passengers always remembering our motto : Everything on time.

In cargo staff page we declared that customer can follow their cargo with using their invoice numbers.



Figure 4.4 Cargo Page

Like transport department the aim of cargo department is that it offers domestic cargo services, has evidence its achievements in the field too, with the concept of safe and swift transport. The fast growing cargo business was organized as a joint stock company. In these days, sg cargo will take its esteemed place in the international arena with its modern automation onset which is unique in the world.

When the customer press the find button a page will appear which asks the invoice number of waybill. After assigning this number system searches the cargo and gives the result to customer.

Before we construct the web site of project we thought that this company has already a web site and these parts can be added to that site. The main idea of us was this. Because this is the most necessary things in a cargo and transportation company.

Finally, if we summarize the whole project starting from first chapter, the concept of project is obtained and the aim of it is realized. Because the main aim of this

project is to comprehend the concept of visual .net and the c# programming language. Also project consist all the required departments and information. For a project the most difficult part is analysis part if you can analysis what ever you do well that time it means you finished the half of project.

CONCLUSION

In this fast growing century the importance of new technology cannot be denied. Every day new systems are inventing and however speed, functionality and harmony of systems are going further.

The main purpose of visual studio.net 2003 supplies all these functions. We selected this technology in our project because of the purposes as we mentioned. Working with .net technology gained us confidence, new imagine techniques and new view angles.

In our project our mission is to create a reliable, useful, fast, harmonious application program. We thought that we almost achieve this. When we look our back, the time period between starting day of our project and this day we can say that we feel our self ready to construct much more better projects.

Our project is compatible with real world business applications. When we look through the real life companies the main concept of project is similar with them. The advantage of this when we faced with real life business applications we would be more familiar.

Finally, the real life is not easy as it seems and to be a part of it we have to work much more and be open minded for new technologies.

REFERENCES

- [1] Building .NET Applications Programming C# Jesse Liberty
- [2] Onion F., Essential ASP.NET with Examples in C#, Addison Wesley, February 11, 2003.
- [3] Beres J., Sams Teach Yourself Visual Studio .NET 2003 in 21 Days, Sams Publishing, January 14, 2003.
- [4] Kennedy B., Musciano C., C#: The Definitive Guide, 5th Edition, O'Reilly Publishing, August 2002.
- [6] Watkins D., Hammond M., Abrams B., Programming in the .NET Environment, Addison Wesley Publishing, November 06, 2002.
- [7] Terence J.J., Payet R.N., ADO .NET Programming, Wordware Publishing, 2003.
- [8] Shepherd G., "DataList vs. DataGrid in C#",
"http://msdn.microsoft.com/msdnmag/issues/01/12/c#/"
- [9] Salman A., "DataSet Made Simple",
"http://www.csharpfriends.com/Articles/getArticle.C#?articleID=2"
- [10] Mitchell S., "Deciding When to Use the DataGrid, DataList or Repeater", 2003.
- [11] A guide for learning C#
"http://www.functionx.com/Community/MessageBoard/Thread.c#?id=270585"
- [12] Manoj R., "Session Management", Retrieved: 03 Mar, 2004
http://www.dotnetspider.com/kb/Article118.c#
- [13] Visual C# .NET başlangıç rehberi İhsan Karagülle