CHAPTER 2

LITERATURE REVIEW

From the beginning to end, my project went through a series of small, but necessary changes. Once I began the programming the code and design interface phase of my design, I came to realize there were quite a few aspects of my design from our original proposal that were simply too ambitious. For example, I was forced to use VB.NET code for design the interface while before I must used VB.NET to design my interface due to time constraints. I also combined the entrance and return interfaces by printing return directions at the entrance interface return as originally proposed.

2.1 What is Visual Studio.NET?

Many people always get confused with Visual Studio .NET (VS.NET) and .NET technology. VS.NET is just an editor, provided by Microsoft to help developers *write .NET programs easily*. VS.NET editor automatically generates lot of code, allows a developer to drag and drop controls to a form, provide short cuts to compile and build the application etc.

Visual Studio .NET add-in that completes code as you write it. Instead of multiline coding constructs such as if..then..else, You just write a 3-letter shortcut. Studio Complete immediately expands the shortcut to the full code. Gain performance by typing less. Works with any .NET language such as VB, C/C++, C#, HTML, XML, XSLT.[4]

2.2 Understand Visual Studio .NET

Visual Studio is a very user friendly tool. But there is enough stuff to confuse any one new to Visual Studio family. The purpose of this chapter is to make me familiar with different options in Visual Studio.NET (VS.NET). You will not cover the entire visual studio guide. You just explaining the most commonly used features of VS.NET.

When you work on *project* system, VS.NET has several child windows to assist you in the application development. These windows are attached on the left, bottom and right sides of the main window. You can click on these small windows to expand it and see the content of them. Some of the most commonly used child windows are explained below. Most of these windows will be enabled only when you have created a project and working on a WinForm. [1]

2.3 What is Visual Basic.NET?

Visual Basic.NET is one of the most widely used an application development tool that has just gotten better. It’s difficult to call Visual Basic.NET the next version of Visual Basic, because both the development environment and the programming language have undergone a nearly complete redesign – one that will challenge millions of programmers to adopt a new programming paradigm.

Visual Basic.NET represents a significant advance in a relatively old programming language. It is this programming language that has evolved over nearly 30 years to become one of the most utilized programming languages in the world. Visual Basic.NET has upgraded some of the features of Visual Basic while introducing some exciting new abilities into the language.

Visual Basic.NET has upgraded some of the old features of Visual Basic while introducing some exciting new abilities into the language, including:

Greater access to system resources

Object inheritance

Garbage collection

Better memory management

Greater interoperability

Common Language Runtime (CLR)

Visual Basic.NET has gotten rid of many older forms of language syntax. This means that programs written in earlier versions (version 6.0 and earlier) may not compile without first being rewritten.

There are some new terms that you should become familiar with. *Managed code* is program code written to run exclusively under CLR. *Unmanaged code* that continuously to rely on the Win32 API and COM. For example, code written in previous versions of Visual Basic is considered unmanaged code. The new Visual Basic.NET compiler creates managed code.

Whether you're new to programming or an experienced Visual Basic programmer, you found that this latest incarnation of Visual Basic is exciting and challenging. While this version maintains a certain amount of the language constructs contained in previous versions, the entire foundation of the language has changed. Visual Basic, once considered "object-based", is now an "object-oriented" programming language.

In addition to object orientation, Visual Basic.NET has other new abilities, such as the ability to create multithreaded applications. The ability to create threads is a basic requirement of some advanced communication abilities, such as network communications.

2.4 Writing Visual Basic.NET Programs

Visual Studio includes a powerful, integrated development environment (IDE). The visual interface allows you to quickly build a user interface by simply selecting controls from a Toolbox window and placing them on a form. Access to the Form and control's properties are only a mouse-click away. Double-clicking the Form or control will add code to handle the control's default event. All that's left for you to do is add the Visual Basic.NET code that provides functionality.

The code editor has many times- saving features, such as auto-formatting, auto-completion, and context sensitive help. Automatic formatting and completion means you're rarely have to worry about a missing **End If** statement or a **Loop** statement in the wrong place. You're rarely had to refer to the documentation to learn what parameters a method accepts, because they will appear in the editor as my type. When the method is overloaded, accepting different parameters, you can scroll to the argument set that you desired.

Also integrated with the editor is the debugger. As you compile the application, warnings and errors are displayed in a Window adjoining the editor. Clicking the warning or error will take me directly to the offending line of code, and normally, a meaningful message will help me fix the problem.

The development environment, Visual Studio, has evolved over the years. The most recent version for the .NET development environment supports an expandable development interface. Visual Studio.NET offers several ways it can be customized. The built-in methods include programmable shortcut keys, tool window configurations, and command bars.

When you're first start Visual Studio.NET, a splash screen will appear. Notice that the splash screen has some useful information. It displays which .NET development products that you had been installed. Make certain that Visual Basic.NET is one of them.

As programmer-friendly as the Visual Studio development environment is, it's not required to write and compile Visual Basic.NET applications. You are free to use any text editor to write the programs, and use the Visual Basic.NET command line utilities to compile the code. Many programmers have formed preferences for code editors, some containing time-saving macros or specific formatting preferences. [4]

2.4.1 Projects

VS.NET allows you to create several types of projects. Most of the time you will be using one of two categories:

- Windows Application - to create any standard windows application.
- ASP.NET Web Application - to create a web site.[1]

2.5 Visual Studio.NET Database

The database project is a special type of Visual Studio.NET project. Its purpose is to create and manage SQL database scripts. If you're developing database applications with Visual Studio.NET, you will want to know about the tools available for making your work with databases easier and faster. This version of Visual Studio features many new tools and contains significant improvements to others that existed in previous versions.

The Visual Database tools allow you to view, design, modify, and test database objects (for example, tables, views, queries, stored procedures, and so on) quickly without having to jump from the Visual Studio environment to a different toolset. The main advantage is in design and development productivity, although licensing and installation issues also have been greatly simplified.

You have already worked with some of these tools, and their use is usually quite intuitive. They should also seem very familiar to me if you had experience using similar tools (even MS Access qualifies). If this is your first time working with such tools, the Visual Studio help topics will be useful.[5]

2.6 Microsoft SQL Server in Visual Basic.NET

A SQL Server database is more complex than an Access database and the actual database engine for SQL Server is made up of multiple components. Also, unlike in Access database, you cannot simply copy a SQL Server database file and distribute it because SQL Server databases consist of multiple files. Procedures must be followed before you can copy and distribute the database files for a SQL Server database.

Those components, other than the database engine and database files, make up SQL Server. These include such components as Replication, Data Transformation Services (DTS), Analysis Services, Meta Data Services, and English Query. SQL Server is a relational database consisting of many components, each of which contains multiple objects.

A SQL Server database consists of at least two files: a data file and a log file. The data file contains all the data that makes up a SQL Server database, such as tables, indexes, and stored procedures. You had examining these objects shortly. The log file contains transaction logs, which are records of changes made to the database.

When you initially create a SQL Server database, the data file and log file are created by SQL Server; the data file has an **.mdf** extension and the log file has an **.ldf** extension. As your database grows and you run out of room on the hard drive, your database administrator may create a secondary data file on a separate hard drive. It will typically have an **.ndf** file extension. Creating a secondary data file for a database typically happens only with large enterprise databases, as most hard drives today can hold multiple database files on a single drive, given their extremely large capacity.[3]

2.6.1 Using a Database

Data-centric applications in this project allow users to view, add, change, and delete data. This data can be textual or binary data. The data used in the system created by this application, entered by the user, and retrieved from archival storage in a database. Of course, the data's source can be any combination of these.

This database can store user information as well as information about itself. You had construct Visual Basic.NET programs that make use of a database depends on the Microsoft SQL Server database[4]

2.6.1.1 Tables

Tables are core objects that exist in the data file and contain information about your project. Each table that you define is made up of columns and rows. Each column represents an attribute about the information stored in your table. Collectively, the columns form a row in your table that represents a single occurrence of the information that the table represents.

Each table in your database usually, but not necessarily, has a column that uniquely identifies each row of data with a *primary key*. No two rows in a table can contain the same primary key, and SQL Server enforces this rule. Primary key columns are usually defined using a *global unique identifier (Guid),* which is a unique value generated based on internal values in your computer. No two computers will ever generate the same unique identifier.[3]

2.6.1.2 Keys

Primary keys may also contain other values such as Totals Amount, which could consist of alpha and numeric characters. Also, primary key columns cannot contain NULL values. A NULL *value* is one that is missing: it does not exist.

When a primary key is created on a table, SQL Server automatically creates a unique index for the primary key on the table. Creating a unique index ensures that no two primary keys can contain the same value. Using the index on the primary key column provides fast, efficient access to the data when using the primary key to access data in a table.

Foreign keys point to the primary key in another table. A foreign key in one row of a table points to an exact row of data in another table. A foreign key value cannot be inserted into a table if the row of data that it is pointing to in another table does not exist. This is just one of the constraints placed on foreign keys that help ensure referential integrity.

Referential integrity enforces the defined relationships between tables when records are inserted or deleted. You cannot insert a foreign key value for a row of data that does not exist in another table. Referential integrity also prevents me from deleting a row of data that is referenced by a foreign key. To do so, you must delete the row of data containing the foreign key or update the column using NULL value. Only then you are able to delete the row containing the primary key.

Referential integrity is based on the relationship between foreign keys and ensures that key values are consistent across all tables. Referential integrity is automatically enforced by SQL Server and prevents a user from updating a primary key in a manner that would break the integrity of the data.[3]

2.6.1.3 Indexes

An *index* is an object associated with tables and is built using one or more columns from a table. An index stores information from columns (usually primary and foreign key columns) and the exact location of that data within the table. Thus, using an index to access information in the table is very efficient, as SQL Server will use the information contained in the index to find the exact location of the row of data that you want retrieve or update. Indexes that are unique do not allow duplicate keys (keys that contain the same data value), and indexes that are not defined as unique can contain duplicate keys. Index keys should not be confused with primary keys in a table. An index key can be generated for any column in a table that is used to access the data in the table.

When making a connection to a database, the first thing you needs is a connection string. This contains the initial catalog, data source name, and optionally the type of security to use. Next, you need to create a connection to the database. This is done using one of the Connection classes (used SQLCommand), passing in the connection string created in the first step.[3]

2.6.2 Using the DataAdapter

The *DataAdapter* forms the bridge between the *DataSet* and the data source (in many cases a database) for the purpose of retrieving and saving data. One of the most powerful methods of the *DataAdapter* is Fill, which sets the contents of the *DataSet* to match data in the data source. The update mehod of the *DataAdapter* calls the appropriate INSERT, UPDATE, or DELETE commands to make changes to the data source. Each row in the *DataSet* is evaluated to see if a new row should be added, an existing row modified, or row deleted.

You add a DataAdapter object for the table. Once again, because you're communicating with an SQL Server database, you use SQLDataAdapter. The

SQLDataAdapter objects acts as a bridge between the DataSet and a Microsoft SQL Server database. It's used together with the SQLCommand object, created in the previous step, and the SQL Command object.

The DataAdapter object has methods that facilitate data storage and editing as well as mapping results to the DataSet:

- DeleteCommand – This property contains the SQL Delete command, used during delete.
- InsertCommand – This property contains the SQL Insert command, used while adding new data.

- SelectCommand – This property contains the SQL Select command, used for retrieving rows.
- TableMappings – This provides the mapping between the DataSet objects and the columns returned from the database.
- UpdateCommand – This property contains the SQL Update command for changing data.

Create a new SQLCommand object by passing in the SQL command as a string. The command object must be told how to interpret the commands given to it as either text or a stored procedure in the database. Set this value to CommandType. Text when not using stored procedures. Otherwise, this property should contain the name of the stored procedure containing the appropriate SQL command. Set the SelectCommand property of the DataAdapter object to the SQLCommand object you just created.[4]

SQL is an important part of modern programming. Whether you're building a Web application, a client-server Windows application, or an enterprise solution encompassing Windows, the Web, and distributed components such as XML Web Services or .NET Remoting, you are probably using a database.

Microsoft has done a great job incorporating tools into Visual Studio .NET, but due to the finite amount of screen real estate; some really great features are tucked away and easy to overlook. SQL points out one of these great features, the SQL designing tools, built right into Visual Studio .NET. You will know how to visually design relationships, filter SQL statements, edit SQL code, and run your queries right in the Visual Studio .NET IDE.

2.6.3 Stored Procedures

A stored procedure is a group of SQL statements compiled into an execution plan and stored under s unique name in the database. It is executed as a unit. A stored procedure can have multiple SQL statements to perform such as tasks as selecting data from one table and updating data in another table.

Stored procedures increase application performance in a couple of ways. First, they enable fewer SQL statements to be transmitted across the network, as you need the name of the stored procedure and any parameters it may require.

Second, stored procedures are similar to procedures and functions in other programming languages, as you can contain input and output parameters and can return values. You use logic to control the flow of processing, and numerous functions and SQL statements can be used in stored procedures.

You can use stored procedures to execute routine functions, such as selecting, inserting, updating, and deleting data. A single stored procedure can be executed by multiple applications, thus providing code reuse.[3]

2.6.3.1 Views

A view is like a virtual table continuing data from one or more tables. A view is stored in the database as the actual SQL statements that are contained in the view, such as a stored procedure. When the view is referenced, the virtual table is created using the SQL statements that are contained in the view.

Views are generally used to enable users to see data from multiple tables in one view, thereby giving the illusion that the data exists as one table or group of data. This provides a couple of benefits. First, by providing the impression that all of the data is in one table, the complexities of the database are hidden from the user. Second, it provides a security mechanism in that you can grant a user access to the view but not to the actual tables from which the view is derived, and you can limit the data a user sees.

Because a view is like a virtual table, you can execute SQL SELECT statements against a view, thereby selecting only the data from the view that you need to see. You can also limit the results by using a SQL Where clause and order the results using a SQL Order By clause.[3]

2.6.3.2 Log files

Each database that you create has its own transaction log. The transaction log contains that have been applied against your database. A transaction is the execution of a group of SQL statements as one logical unit of work. SQL Server automatically manages transactions in the transaction log, generating a before-and-after picture of the data in a table that is changed. This means that you can execute an update query to update a row of data and SQL Server logs a record of the data in your database.

SQL Server manages transaction logging automatically. You can, however, use transactions in your stored procedures to perform automatic recovery of the data that you stored procedures changed. Your can also use transactions in the ADO.NET classes that provide data access to my database. [3]

2.6.4 Making a Database Connection

When making a connection to a database, the first thing you will need is a connection string. This contains the initial catalog, data source, or data source name, and optionally the type of security to use. Next, you will need to create a connection to the database. This is done using one of the *Connection* classes (use SQLCommand), passing in the connection string created in the first step. [4]