CHAPTER 3

METHODOLOGY

3.1 Methodology Review

In the Development of Smart Parking System, I have chosen the simplest rendition of this system. It is called the “waterfall model”. The waterfall model is a sequential software development model (a process for the creation of software) in which development is seen as flowing steadily downwards (like a waterfall) through the phases of requirements analysis, design, implementation, testing (validation), integration, and maintenance. Below is the unmodified "waterfall model". Progress flows from the top to the bottom, like a waterfall.

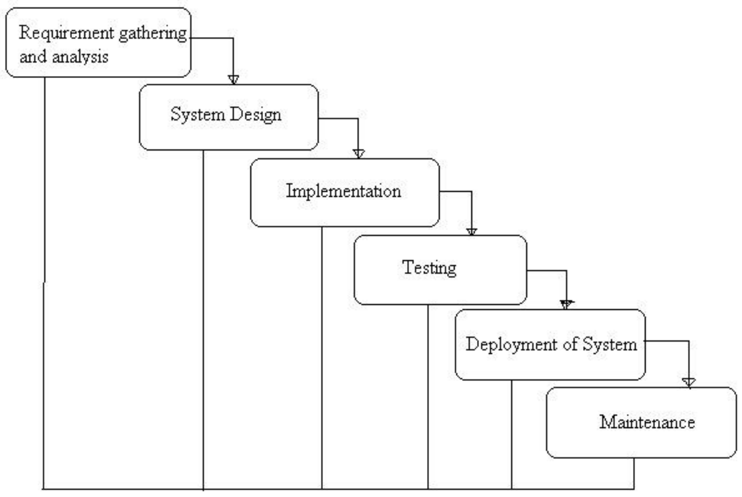


Figure 3.1: Waterfall Model

3.2 PHASE 1: System Requirements Analysis

All possible requirements of the Development of Smart Parking System to be developed are captured in this phase. Requirements are set of functionalities and constraints that the end-user (who will be using the system) expects from the system. The requirements are gathered from the end-user by consultation, these requirements are analyzed for their validity and the possibility of incorporating the requirements in the system to be development is also studied. Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model.

This is also known as feasibility study. In this phase, I must visit the customer and studies their system. I had investigating the need for possible software automation in the given system. By the end of the feasibility study, I had furnishes a document that holds the different specific recommendations for the system. It also includes the project schedule, and target dates. The requirements gathering process is intensified and focused specially on software. To understand the nature of the program(s) to be built, I must understand the information domain for the software, as well as required function, behavior, performance and interfacing. The essential purpose of this phase is to find the need and to define the problem that needs to be solved.

Planning is to establishing the plans for creating an information system by defining the system to be developed; a system must be identified and chosen. The project scope is a high level of system requirements must be defined and put into a project scope document. Developing the project plan needs all details from tasks to be completed, which completed them and when I were completed must be formalized. Managing and monitoring the project plan can allows the organization to stay on track, creating project milestones and feature creeps which allow me to add to the initial plan

Analysis for the users and IT specialists collaborate to collect, comprehends, and logistically formalize business requirements by gathering the business requirements.

3.3 PHASE 2: System & Software Design

In this phase, the software development process, the software's overall structure and its nuances are defined. In terms of the client/server technology, the number of tiers needed for the package architecture, the database design, the data structure design is all defined in this phase. A software development model is created. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

Before a starting for actual coding, it is highly important to understand what I am going to create and what it should look like? The requirement specifications from first phase are studied in this phase and system design is prepared. System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture. The system design specifications serve as input for the next phase of the model.

Design is where the technical of the system is created by. Designing the technical architecture by choosing amongst the architectural designs of hardware and software that will best suit the organization’s system and future needs. Designing the systems model is graphically creating a model from graphical user interface (GUI), GUI screen design, and databases, to placement of objects on screen.

Development is executing the design into a physical system by. Building the technical architecture for purchasing the material needed to build the system. Building the database and programs is the IT specialists write programs which will be used on the system.

3.3.1 Block Diagram: Parking Garage Schematic

The block diagram, shown in Figure 3.2, displays the basic organization and flow of data between the various components of my project. A more detailed explanation of each block will be presented in the following report. All the system in this block diagram using VB.NET interface.

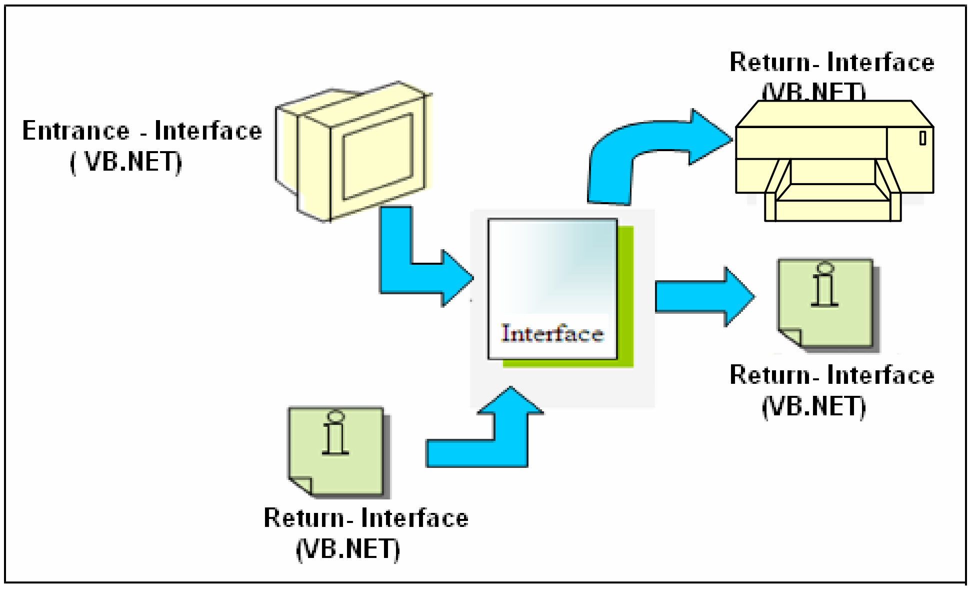


Figure 3.2: Parking Garage Schematic

3.3.2 Parking Garage Schematic Descriptions

3.3.2.1 Entrance-Interface

The entrance-interface, programmed in VB.NET, consists of a monitor and mouse that allows the driver to choose their preferred parking spot. Parking spot information will be sent to the next interface. The interface is the display that the parking driver will see.

3.3.2.2 Interfaces

The timer displays the time remaining or indicates if time has run out in the interface. This is already built into the parking system. The database detects the presence of a vehicle. The output is sent to the interface.

3.3.2.3 Return-Interface

The Return-Interface is the display parking attendance will see. The display will indicate the car number, total amount and the length of time of the parked vehicle. It also programmed in VB.NET, consists of a monitor and mouse that allows the parking attendance to enter their parking spot number upon returning to the garage. The display will indicate the car number, the length of time and the totals amount of the parked vehicle.

3.3.3 Flowchart Parking Garage

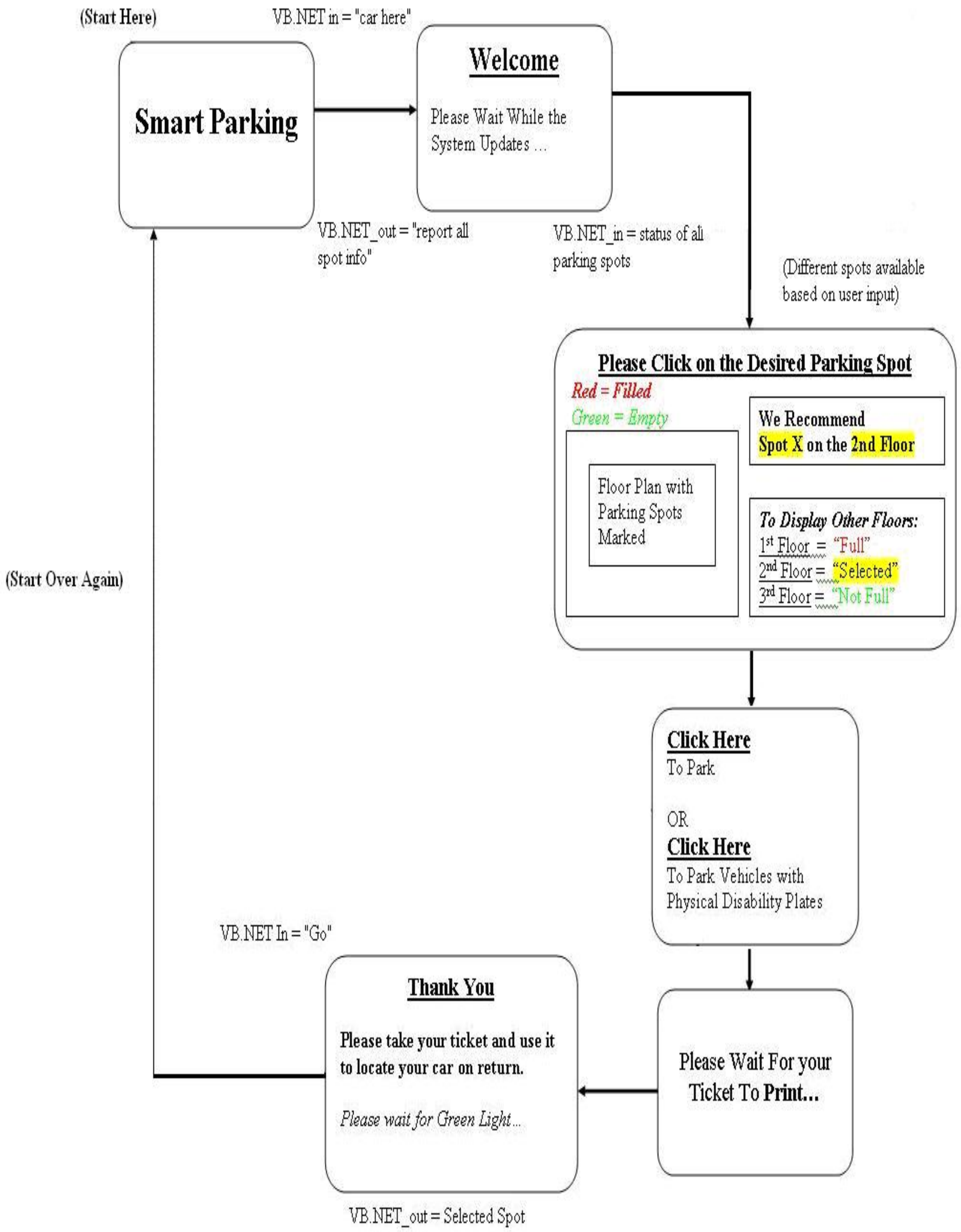

(Start Here)
VB.NET in = "car here"
Smart Parking
Welcome
Please Wait While the System Updates ...
VB.NET_out = "report all spot info"
VB.NET_in = status of all parking spots
(Different spots available based on user input)
Please Click on the Desired Parking Spot
Red = Filled
Green = Empty
Floor Plan with Parking Spots Marked
We Recommend Spot X on the 2nd Floor
To Display Other Floors:
1st Floor = "Full"
2nd Floor = "Selected"
3rd Floor = "Not Full"
(Start Over Again)
Click Here To Park
OR
Click Here To Park Vehicles with Physical Disability Plates
VB.NET In = "Go"
Thank You
Please take your ticket and use it to locate your car on return.
Please wait for Green Light...
Please Wait For your Ticket To Print...
VB.NET_out = Selected Spot

Figure 3.3: Flowchart Development of Smart Parking System

3.4 Performance Requirement

Since my project was a computer program, testing included debugging the code and running through the program to make sure it did what was intended. The proper performance of the system can be easily monitored by VB.NET interfaces and Microsoft SQL Sever database observation to interfacing them. I had to check to make sure it not only found the parking lot, but also found the closest parking lot to that system. Real time acquisition is my initial idea is to transmit updates (on the presence of a car) every minute.

This was accomplished by first only having one free parking spot available on the entire map. The program of course found that parking lot, but I manually checked to make sure the system that came up was the shortest program. Once this was successful, I increased the number of free parking lots and again checked by hand the result of the program. At fourth free parking lots on the map, I felt satisfied that the program worked properly and felt that any more testing would be too time consuming and would probably not come up with any errors.

The main limitation on operating conditions for this program is that the source code finding routine is a recursive function that crashes the system at times. A recursive program makes exponential calls to it, and takes up a lot of system memory. The error that I receive is a “out of stack space” error that is a limitation of the system and the Visual Basic.NET programming environment.

This lack of true consistency in the program is unexpected and because of time requirements, I am not able to fix it. However, I still feel confident that the simulation gives a user a good “proof of concept” to demonstrate the fully working system.

3.4 PHASE 3: Program

The design must be translated into a machine-readable form. The code generation step performs this task. If the design is performed in a detailed manner, code generation can be accomplished without much complication. Different high level programming languages like VB.NET and Microsoft SQL Server are used for coding. With respect to the type of application, the right programming language is chosen.

3.4.1 VB.NET Interface Programming

I decided to program the entrance interface in VB.NET mainly for its graphical environment and ease of use. Alternatively, the interface could have been programmed in JAVA, C++ or any other object oriented programming language.

I organized the VB.NET program by interface screen. Each button on the screen masks a function that is executed when the button is selected. To send and receive information I used pre-defined functions within the MSCOMM library in VB.NET.

3.4.1.1 Getting Started with VB.NET

To complete the steps, I must first install Visual Studio.NET on my laptop. Then I am ready to start working with VB.NET. To start VB.NET in Windows XP Professional:

1. Click the Start button, point to All Programs, point to Microsoft Visual Studio.NET 2003, and then click Microsoft Visual Studio.NET 2003.

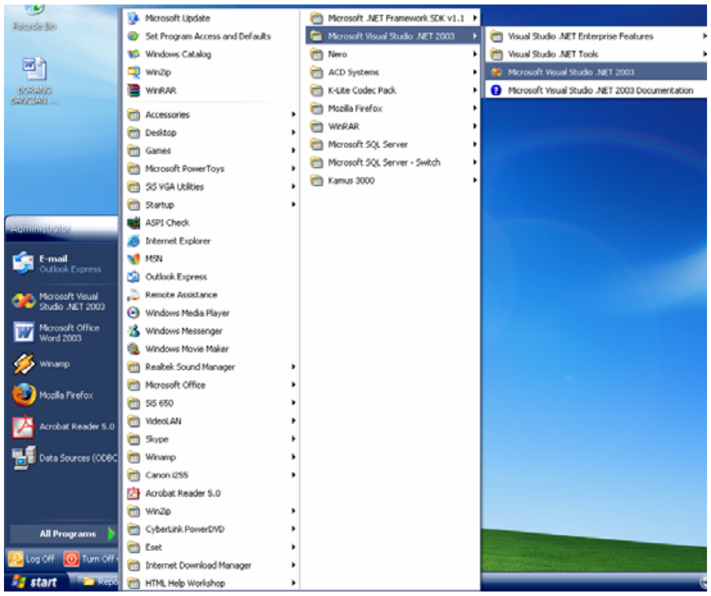
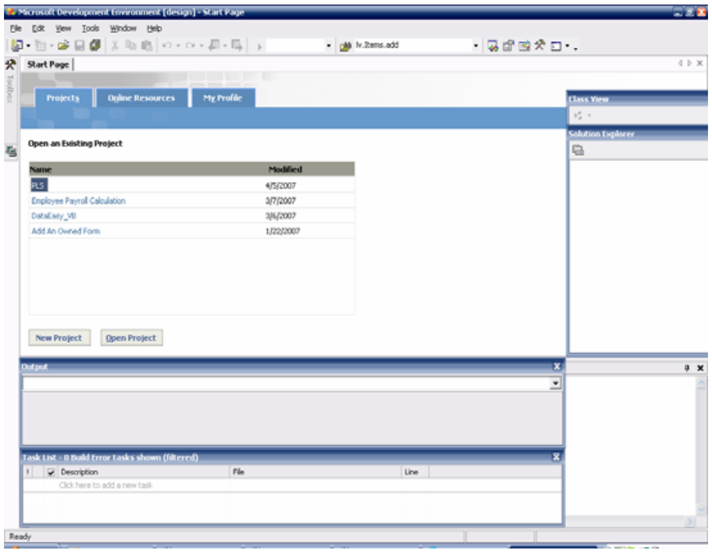


Figure 3.4: Starting Visual Studio.NET

2. A splash screen identifying the Visual Studio.NET languages installed on my laptop momentarily appears, and then I see the Microsoft Development Environment (MDE) window. Verify that the MDE window resembles.(My screen may vary slightly depending on the settings for my computer).


Microsoft Development Environment [design] - Start Page
File Edit View Tools Window Help
Start Page
Projects
Online Resources
My Profile
Class View
Solution Explorer
New Project
Open Project
Output
Task List - 0 Build Error tasks shown (filtered)
Description
File
Line
Click here to add a new task
Ready

Open an Existing Project

Name	Modified
PLS	4/5/2007
Employee Payroll Calculation	3/7/2007
DataEasy_VB	3/6/2007
Add An Owned Form	1/22/2007

Figure 3.5: The Microsoft Development Environment

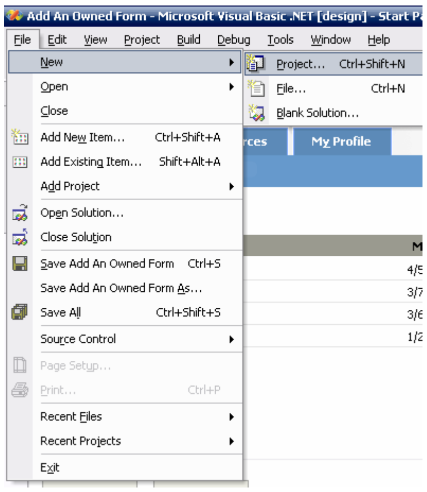
3.4.1.2 Creating a Windows Application

I will now create VB.NET project for a Windows application. I use the Windows Form Designer to create an input form for the application. Recall that when I create a project, I must identify the project type and template I want to use, and specify the project name and location. To create a Windows application:

1. If I have not already done so, start Visual Studio.NET.

2. If is not already selected, click the **Start Page** tab in the document window, and the click the **Projects** tab.

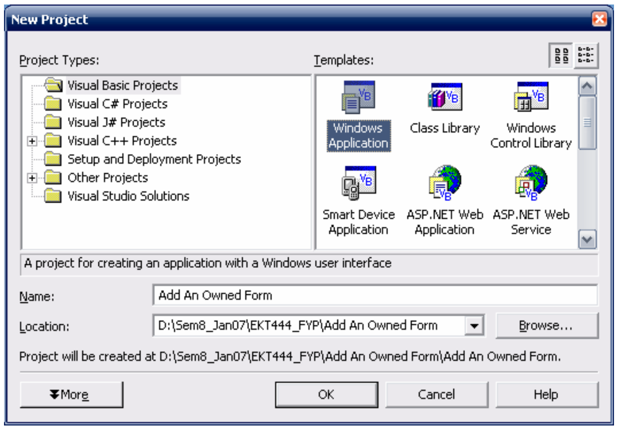
3. Click the New Project button. The New Project dialog box opens.



Add An Owned Form - Microsoft Visual Basic .NET [design] - Start P
File Edit View Project Build Debug Tools Window Help
New
Open
Close
Add New Item... Ctrl+Shift+A
Add Existing Item... Shift+Alt+A
Add Project
Open Solution...
Close Solution
Save Add An Owned Form Ctrl+S
Save Add An Owned Form As...
Save All Ctrl+Shift+S
Source Control
Page Setup...
Print... Ctrl+P
Recent Files
Recent Projects
Exit
Project... Ctrl+Shift+N
File... Ctrl+N
Blank Solution...
My Profile

Figure 3.6: File > New > Project

4. In the New Project dialog box, click the Visual Basic Projects folder in the Project Types pane. Click the Windows Application icon in the Templates pane.


New Project

Project Types:
- Visual Basic Projects
- Visual C# Projects
- Visual J# Projects
- Visual C++ Projects
- Setup and Deployment Projects
- Other Projects
- Visual Studio Solutions

Templates:
- Windows Application
- Class Library
- Windows Control Library
- Smart Device Application
- ASP.NET Web Application
- ASP.NET Web Service

A project for creating an application with a Windows user interface

Name: Add An Owned Form

Location: D:\Sem8_Jan07\EKT444_FYP\Add An Owned Form

Project will be created at D:\Sem8_Jan07\EKT444_FYP\Add An Owned Form\Add An Owned Form.

Figure 3.7: The new Project dialog box

5. The Name text box will contain the default project name Windows Application1. I delete the default project name and type Add an Owned Form in the Name text box.

6. In the Location text box, specify the parent folder for my project. I can accept the default location or use the Browse button to select an alternate location.

7. Click OK. The Windows Form Designer appears as a tabbed document labeled Form1.vb [Design]. By default, the Windows Form Designer names files that contain forms as Form1.vb, Form2.vb, and so on.

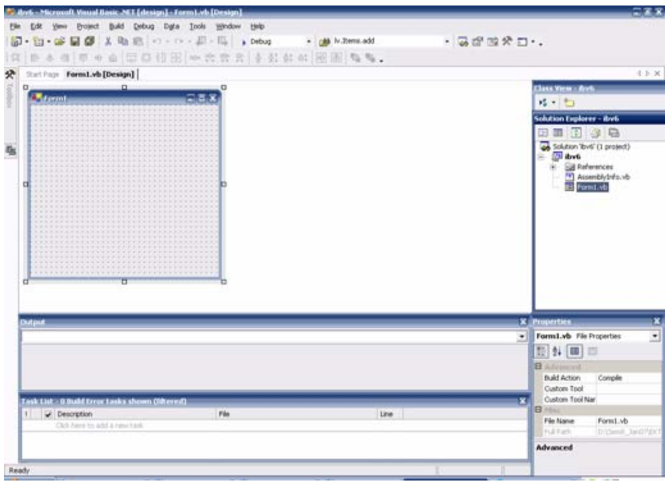


Figure 3.8: The Windows Form Designer

3.4.1.2(a) Toolbox

Toolbox provides all the drag and drop controls for my application. Depending on the kind of application I'm working on, the toolbox show appropriate controls and I can drag and drop them to my form. I'm developing a Windows application; it will show Windows controls (like Radio button, text box, buttons etc).

I can simply drag and drop any controls from the toolbox to my form. After I drag and drop any control, double click on the control to go the default event associated with the control.

The Toolbox will be enabled only when I have a WinForm opened in *Design mode* and am usually located on TOP LEFT corner of the VS.NET. By default, all windows including Toolbox will be displayed as *Minimized* (*Marked as An in picture*). I will see only the small icon and the text 'Toolbox' written vertically on the left bar of VS.NET. I can click on this minimized window to expand it (*Expanded window is marked as C.*). When I move the mouse away from the window, it will again automatically minimize. I can keep the toolbox always expanded by pressing the *pushpin* (*Marked with B.*). The above behavior is common for all the windows explained below. They will be minimized by default and I can point the mouse over it to expand it. Use the pushbutton to keep it expanded.

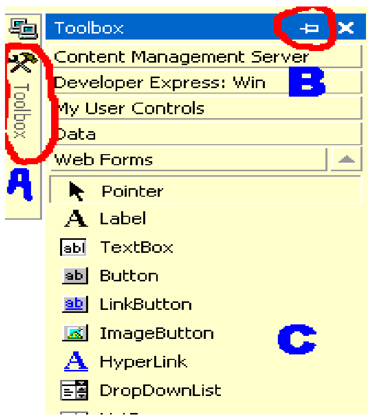


Figure 3.9: Toolbox

3.4.1.2(b) Solution Explorer

Solution Explorer, located on TOP RIGHT corner of VS.NET, displays my solution, all projects included in each solution and the list of files in each project. They are listed in the form of a tree control.

Typically, for a single application, I have one solution and one or more projects. When I create a new application, I have to create single Visual Studio project. In more complex applications, there may be more than one project. All these projects are grouped into a single solution. Even if I do not create a solution separately, a solution will be automatically created for me.

To add a new file to my project, I can right click on the project name (**Add An Owned Form** is the project name in the picture) and choose the menu option **Add**. It will give me the option to choose a file type. I can choose an appropriate type.

For WinForms, I can see the file in design mode and the code associated with it. Double click on any form and it will be opened in design mode. Right click on any form and select the menu option 'View Code' to view the code associate with the form.

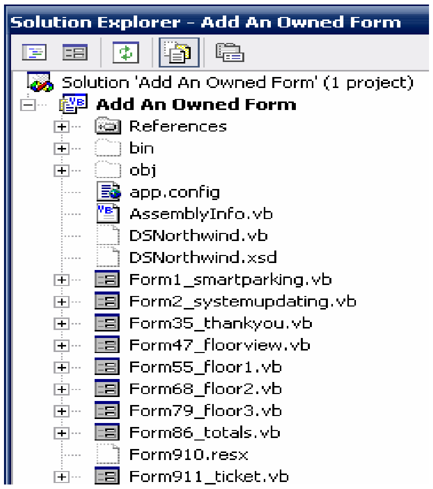


Figure 3.10: Solution Explorer

3.4.1.2(c) Output

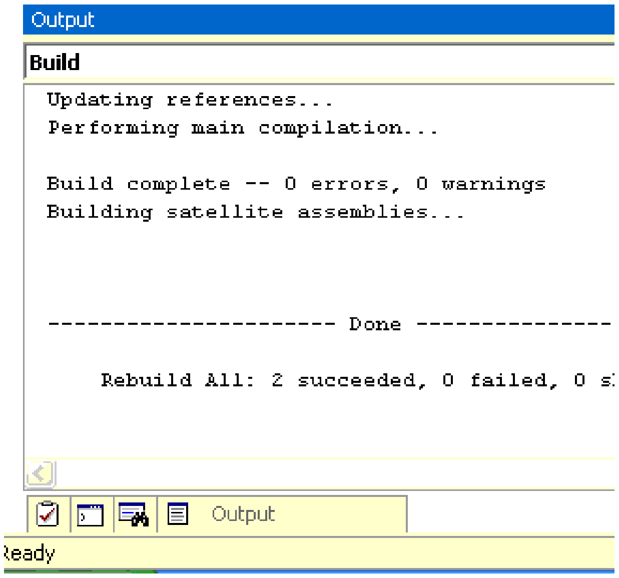
Output window, located on BOTTOM LEFT corner of VS.NET, displays the result of my project compilation. When I compile my project, all errors, warnings and compilation results will be displayed in this window.

In addition to the Output window, they are few other windows located in the BOTTOM LEFT corner of VS.NET.

- **Task List** -shows individual tasks. Typically, when I compile my project, all errors and warnings will be added to my task list. I can double click on any item in the task list to go directly to the code associated with the task.

- **Command Window** – I can execute code statements here. When I'm debugging, I can evaluate the value of any variables by typing? Followed by the variable name.

- **Find Results** - when I search for any text in file(s) using VS.NET, the results will be displayed in this window.



```
Updating references...
Performing main compilation...

Build complete -- 0 errors, 0 warnings
Building satellite assemblies...



---------------------- Done ---------------

    Rebuild All: 2 succeeded, 0 failed, 0 s
```

Figure 3.11: Output

3.4.1.2(d) Building a project

To try this, create a *Windows Project* as explained above. It will create a sample form. Go to the main menu and select the menu item **Build > Build Solution**.

This process will compile all the files included in my project and show me the result in the *Output* window. If the result shows '0 failed', my build is success and my application is ready to deliver!!

To *Run* the application I just *Built*, go to the main menu and select **Debug > Start Without Debugging**. This will launch the application that I just developed. I can drag and drop several controls to the form and try running it.

When I compile (build) the code, if there is any errors or warnings, the details will be shown in the 'Task List' window. I can click on the specific item in this window to go directly to the line of code associated with the error.[2]

3.4.2 Visual Studio.NET Database

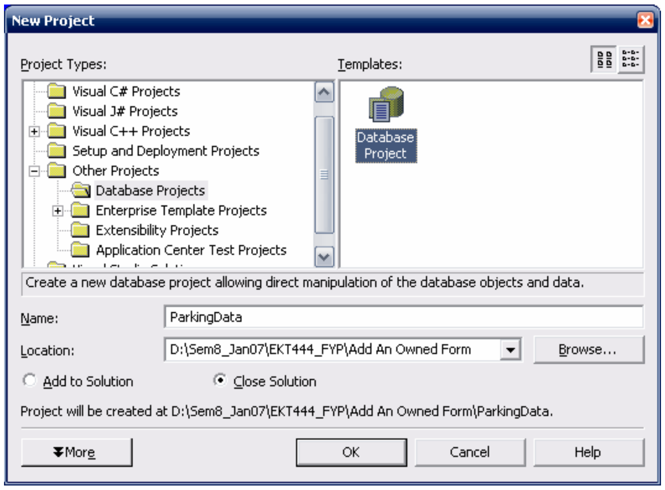
3.4.2.1 Creating a Database Project

A special type of VS.NET project of particular interest is the database project. A database project is not specific to any .NET programming language, such as Visual Basic or C#, but is meant to be used to design, test, run, and manage SQL scripts and queries. When I design my application in multiple layers, as I should, this project type helps me to develop and manage the database layer closest to the database and the database itself.

In some applications, a database project may be part of the actual application code, and in other applications, it may be part of a separate administration solution used to set up and maintain the application's database.

To add a new database project to a solution, do the following.

1. Launch the Add New Project dialog from either the main File menu or from the context menu displayed by right-clicking on the solution in the Solution Explorer.
2. The left panel of this dialog box displays a list of folders containing different project types. Expand the Other Projects folder (click on the "+").
3. Select the Database Projects folder. It displays a Database Project template in the right panel of the dialog box.
4. Specify a project name of ParkingData and a path for saving the project files and then click on the OK button.



Create a new database project allowing direct manipulation of the database objects and data.

Name: ParkingData

Location: D:\Sem8_Jan07\EKT444_FYP\Add An Owned Form

Add to Solution [✓ Close Solution]

Project will be created at D:\Sem8_Jan07\EKT444_FYP\Add An Owned Form\ParkingData.

Figure 3.12: New Database Project

5. If I don't currently have any database connections defined in the Server Explorer; the Data Link Properties dialog box will be displayed, allowing me to define a new database connection.
6. If I have at least one database connection defined in the Server Explorer, the Add Database Reference dialog box is displayed. From this dialog choose the database connection that want to use (I can add new ones later). Alternatively, I can click on

the Add New Reference button to display the Data Link Properties dialog box, allowing me to define a new database connection.

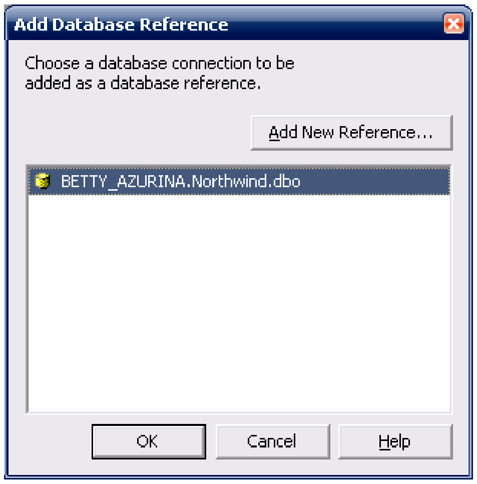

Add Database Reference
Choose a database connection to be added as a database reference.
Add New Reference...
BETTY_AZURINA.Northwind.dbo
OK
Cancel
Help

Figure 3.13: Database connection

7. Select the connection to the ParkingData database on the SQL Server and click on the OK button. If for some reason the connection doesn't exist, create it from the Data Link Properties dialog box.
8. Figure shows the project and its folders in the Solution Explorer.

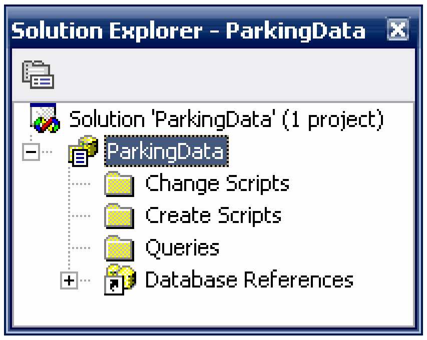

Solution Explorer - ParkingData
Solution 'ParkingData' (1 project)
ParkingData
Change Scripts
Create Scripts
Queries
Database References

Figure 3.14: The ParkingData database project shown in the Solution Explorer

Note that the created database project contains the following folders:

- Change Scripts
- Create Scripts
- Queries
- Database References

3.4.3 Managing SQL in Visual Studio.NET

3.4.3.1 Connection in the Server Explorer

The Server Explorer - accessible from View>Server Explorer - is a focus point for several tools tucked away into Visual Studio .NET. The Data Connections node contains OLEDB database connections, and the Servers node (see Figure) contains a SQL Servers child node. If I have a network connection or a local copy of SQL Server, such as MSDE, those database servers will show up here.

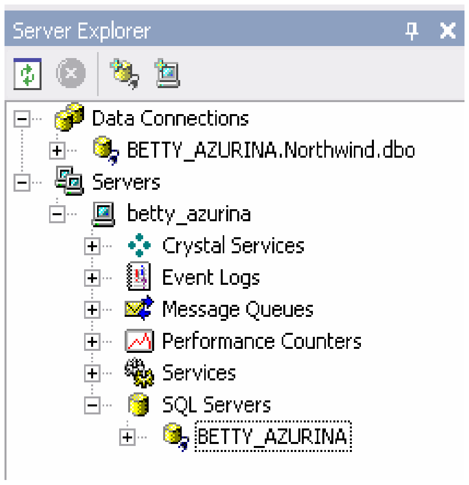
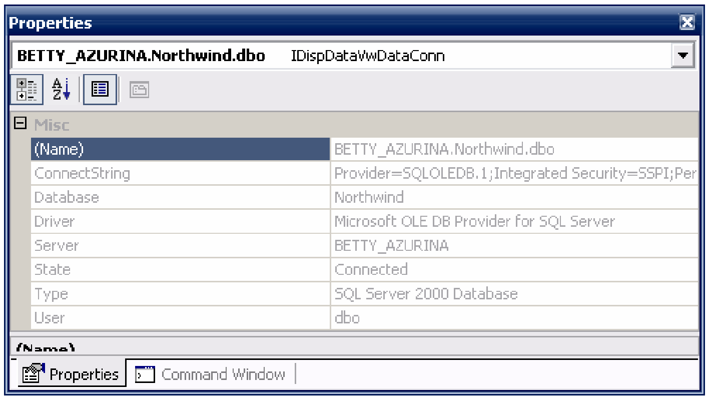


Figure 3.15: The Data Connections and SQL Servers nodes in the Server Explorer.

If I right-click on the Data Connections node, I can select Add Connection or Create SQL Server Database to connect to an existing database or create a new database. If I right-click on the SQL Servers node, I can register a SQL Server instance in Visual Studio .NET. Select any of these databases, press F4, and the Properties window will provide me with detailed information about the particular database. Of particular use is the ConnectString property that will provide me with convenient access to what can oft be a cryptic string of connection information. For example, if I am working with ADO.NET and need a connection string to initialize a connection object, I can copy a correct connection string from the Properties window. [7]



(Name)	BETTY_AZURINA.Northwind.dbo
ConnectString	Provider=SQLOLEDB.1;Integrated Security=SSPI;Per
Database	Northwind
Driver	Microsoft OLE DB Provider for SQL Server
Server	BETTY_AZURINA
State	Connected
Type	SQL Server 2000 Database
User	dbo

Figure 3.16: The Properties for a database instance in the Server Explorer.

3.4.3.2 Browsing Tables, Views, and Stored Procedures

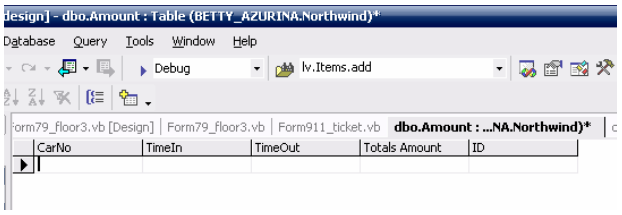
If individual Data Connections or SQL Server instances in the Server Explorer are expanded, the context menus for those items and those sub-nodes permit me to create and explore tables, views, and stored procedures. Thus, I can still use the SQL Enterprise Manager or MS-Access (or whatever database tool you generally use) to manage data, but I don't have to leave Visual Studio .NET for most of tasks.

Occasional exceptions to the general utility of Visual Studio .NET for managing databases can be managed by falling back on the tools shipped by the database provider. For instance, I can write and test stored procedures for SQL Server in Visual Studio .NET, but I need to use the stored procedure builder that ships with UDB to implement stored procedures for a UDB database. (Although I can run stored procedures in Visual Studio .NET, current implementations don't permit editing stored procedures supported by non-SQL Server providers.).[7]

3.4.3.3 Viewing the Data

I use the Northwind sample database for my examples because of its ubiquity. If I don't have a sample database listed in the Data Connections node, use the Add Connection feature to add a database connection.

The easiest way to view data is to expand the database and its Tables node. Double-click on the table whose data I‘d like to view and a datagrid view corresponding to a SELECT * FROM *tablename* is displayed (see Figure 3.17). In addition to showing the data, the Query toolbar is displayed - represented by the twelve buttons on the bottom of the toolbar (the third one from the left contains the acronym SQL).


design] - dbo.Amount : Table (BETTY_AZURINA.Northwind)*
Database Query Tools Window Help
Debug
lv.Items.add
Form79_floor3.vb [Design] | Form79_floor3.vb | Form911_ticket.vb | dbo.Amount : ...NA.Northwind)*

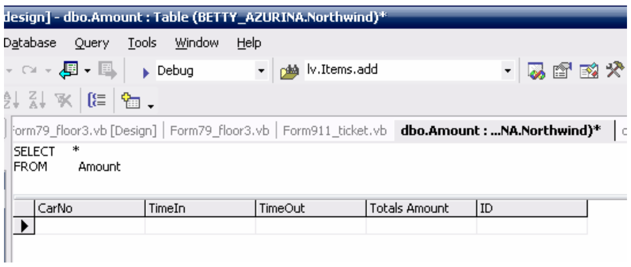
CarNo	TimeIn	TimeOut	Totals Amount	ID

Figure 3.17: The center shows the Results Pane controlled by the fourth Query button from the left.

The context menu for the Results Pane contains options for navigating through the results shown. In addition to the context menu, the Query menu contains additional features for managing the data in the Results Pane and performing other database tasks. For example, to create a new table from the existing table in the Results Pane, select Query|Change Type|Make Table and the Results Pane's query will be modified to include an INTO clause that will cause results from the current view to be copied into a new table named by me. In essence, I can create a copy of the current table. To view the default query or the modified make table query, click the SQL button from the Query toolbar to display the SQL Pane.[7]

3.4.3.4 Writing SQL

The combined SQL Pane and Results Pane containing the default SELECT statement and the results of that SELECT statement are shown in Figure 3.17. The SQL Pane is just an editor. I can write any valid SQL in this editor, and save and load this SQL at any time to and from an external file. In addition, Visual Studio .NET is a great first-line tool for managing data during development and unit testing.


design] - dbo.Amount : Table (BETTY_AZURINA.Northwind)*
Database
Query
Tools
Window
Help
Debug
lv.Items.add
Form79_floor3.vb [Design]
Form79_floor3.vb
Form911_ticket.vb
dbo.Amount : ...NA.Northwind)*

```
SELECT    *
FROM      Amount
```

CarNo	TimeIn	TimeOut	Totals Amount	ID

Figure 3.18: The SQL Pane shown top and the Results Pane shown bottom, demonstrating the coordination between SQL code and related results.

When I have written code, I can change the type of the query from the Change Type toolbar dropdown list. I can run the SQL from the Run Query toolbar button (red exclamation mark), verify the syntax of the current query (with the SQL button with a checkmark), and a GROUP BY clause (second button from the far right) and add additional tables to the query. (I'll appreciate the Add Table because it will write syntactically correct join statements. [7]

3.5 PHASE 4: Verification

Having a fully working system, that demonstrates the objective of my project. This will be done in phases and each phase would be able to be tested independent of the other phases.

3.5.1 Testing Procedures

- Each component will be tested individually before integration.
    - Compile and run VB.NET program
    - Testing the interface for all user input possibilities
    - Testing the connection database for all user input possibilities

  Test integrated system as each feature is added. First, I will test assuming the ideal case (one car). Next, I will simulate and test the system's capabilities for guiding multiple cars and pedestrians in the garage.

- To test the Microsoft SQL Server, I will input a password. The password will be sent to the computer.

- Compiling and running software, which displays the location of the available parking spots.

- Compiling and running software (on a PC), which would store, organize, and process the database.

- To test the interface receiver from multiple interfaces, I will send to know password from server, and verify that the same password will received at the interface.

- To test the overall system, I will park three fictional cars at three fictional parking spaces, and see if the location and the amount of time illegally parked indeed show up at the interface.

- Testing and running a fully working system that is implemented in real-life.

3.5.2 Tolerance Analysis

- The system should be capable of handling the ideal scenario, which is when there is only one car in the garage.

- For the Microsoft SQL Server database, firstly, it must install the driver before connect to the server and and database.

- The system is addendums to the project which may or may not be implemented due to time constraints and level of difficulty.

- After all individual components were tested and verified, the VB.NET interfaces were integrated with the Microsft SQL Server Database for a complete, fully functional as Smart Parking System.

- The interfaces must be accurate enough to detect the car only within its designated parking area.

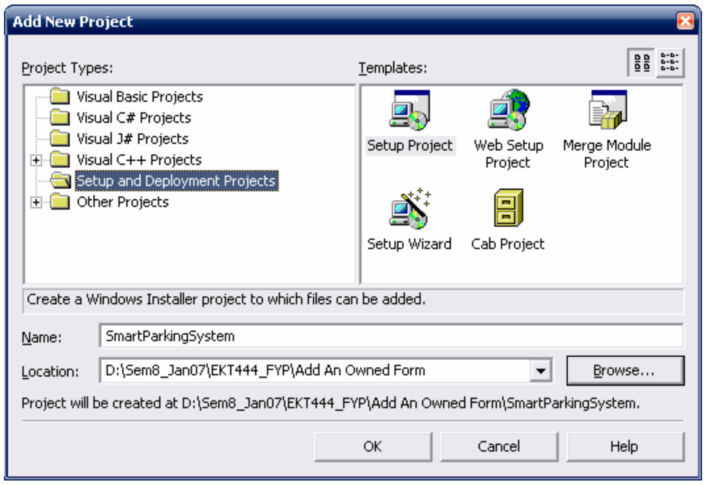
- The component that would most affect this project would be the Microsoft SQL Server. This depends on the interfacing of the database. The database must be precise enough to detect the car number only within its designated parking spot. This includes the transmission of interface form the database to the central computer

3.6 PHASE 5: Deploying Application

3.6.1 Adding the Deployment Project

Now that my application is coded and running fine, I need to think how I am going to distribute the application. Open the solution explorer, right click the solution and select Add, then New Project. The same action could be achieved from the File menu as well.

In the Add New Project window, select Setup and Deployment Projects from the project types. Also, select Setup Project from the Templates list. Change the name to SmartParkingSystem, and select the location where I want to store your project file(s). Change this location to the main application project folder. This will ensure that all project and setup file(s) are created in folders under your main project folder. [6]



Create a Windows Installer project to which files can be added.

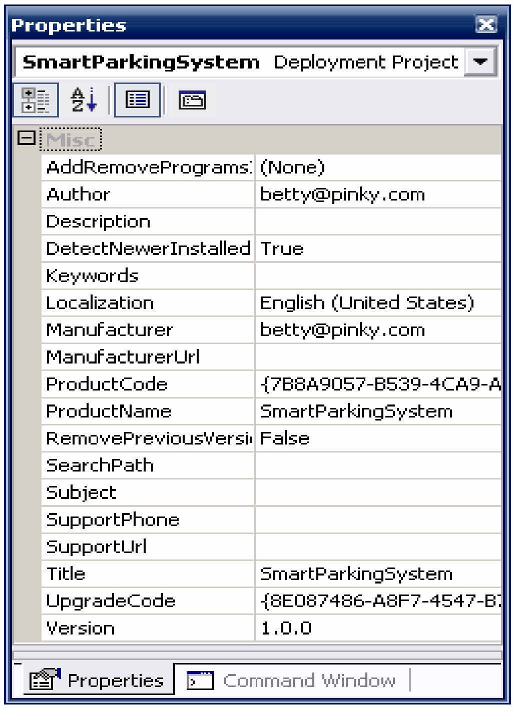
Name: SmartParkingSystem

Location: D:\Sem8_Jan07\EKT444_FYP\Add An Owned Form

Project will be created at D:\Sem8_Jan07\EKT444_FYP\Add An Owned Form\SmartParkingSystem.

Figure 3.19: Setup and Deployment Projects

Click OK. The Setup and Deployment project will be added to my existing project. Clicking OK will also open the File System window (we will go through that in a moment). Select the SmartParkingSystem project from the Solution Explorer and open the properties window. I can change the following properties:

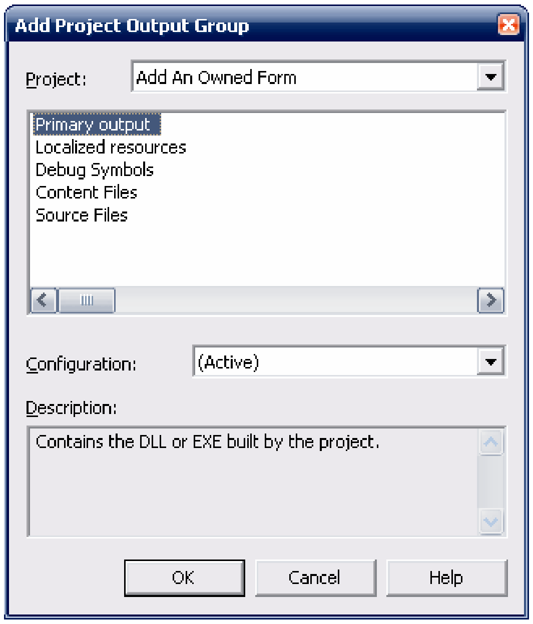


AddRemovePrograms:	(None)
Author	betty@pinky.com
Description	
DetectNewerInstalled	True
Keywords	
Localization	English (United States)
Manufacturer	betty@pinky.com
ManufacturerUrl	
ProductCode	{7B8A9057-B539-4CA9-A
ProductName	SmartParkingSystem
RemovePreviousVersi	False
SearchPath	
Subject	
SupportPhone	
SupportUrl	
Title	SmartParkingSystem
UpgradeCode	{8E087486-A8F7-4547-B
Version	1.0.0

Figure 3.20: SmartParkingSystem Deployment Project Properties

Open the Solution Explorer, right click on the SmartParkingSystem project, click View and select File System to open the File System window.

From the File System on the Target Machine, right click the *Application Folder*, click Add and select Project Output. In the Add Project Output Group, select Primary output from the list of options available and finally, click OK.



Add Project Output Group

Project: Add An Owned Form

Primary output
Localized resources
Debug Symbols
Content Files
Source Files

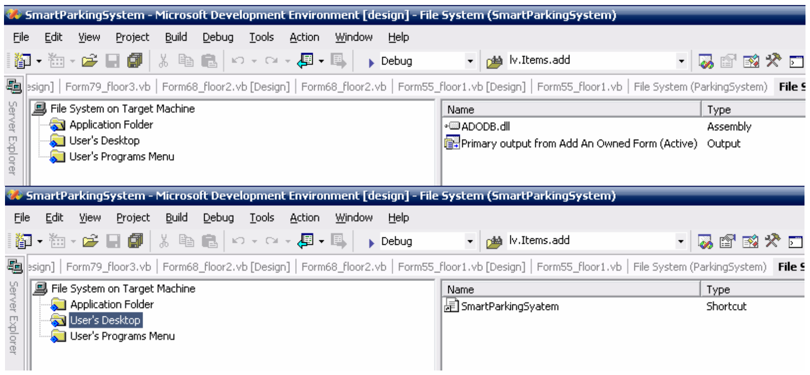
Configuration: (Active)

Description:

Contains the DLL or EXE built by the project.

Figure 3.21: Select Primary Output from Project Output Group

To run the application on the target machine, we need to create a shortcut for the application on my desktop. To do this, right click on the Primary Output from Add An Owned Form (Active) and select Create ShortCut to Primary Output from Add An Owned Form (Active). Rename this newly created entry to SmartParkingSystem and drag this entry (SmartParkingSystem) to the user's Desktop in the File System on Target Machine Window. The screenshot below shows me what everything should look like:



SmartParkingSystem - Microsoft Development Environment [design] - File System (SmartParkingSystem)

File
Edit
View
Project
Build
Debug
Tools
Action
Window
Help
Debug
lv.Items.add
esign] | Form79_floor3.vb | Form68_floor2.vb [Design] | Form68_floor2.vb | Form55_floor1.vb [Design] | Form55_floor1.vb | File System (ParkingSystem) | File S
Server Explorer

File System on Target Machine
Application Folder
User's Desktop
User's Programs Menu

Name	Type
ADODB.dll	Assembly
Primary output from Add An Owned Form (Active)	Output

SmartParkingSystem - Microsoft Development Environment [design] - File System (SmartParkingSystem)

File System on Target Machine
Application Folder
User's Desktop
User's Programs Menu

Name	Type
SmartParkingSyatem	Shortcut

Figure 3.22: Primary Output SmartParkingSystem

3.6.2 Building the Deployment Project

Open the Solution Explorer and select the SmartParkingSystem project. From the Build menu, select Build SmartParkingSystem to build my deployment project. This procedure may take a few minutes, depending upon my machine and will create a Debug folder under the SmartParkingSystem folder. [6]

3.6.2.1 Looking at Installer Files

After the Setup Project is built without any errors, the following files will be created:

- InstMsiA.exe
- InstMsiW.exe
- SmartParkingSystem.msi
- Setup.exe
- Setup.Ini

SmartParkingSystem.msi is the Windows Installer file for my application. Setup.exe is the wrapper for the SmartParkingSystem.msi, which checks if the correct version of Windows installer is available before the application is installed.

InstMsiA.exe and InstMsiW.exe is the Windows Installer application file(s), which are used to support different Windows platforms, such as Windows 95, Windows 98, Windows ME, Windows NT and Windows 2000.

I had made sure to distribute all of these files when I deploy my application and always run Setup.exe to install the application. I've finished creating my deployment application! [6]

3.6.3 Installing the Application

To test and run my installer, click Install from the Project menu. Alternatively, I could also start the installation by running setup.exe from the installation folder.

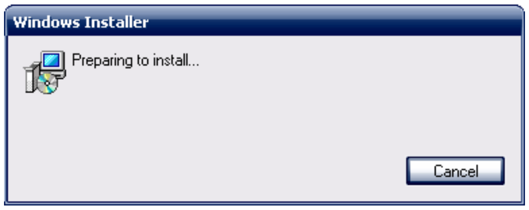
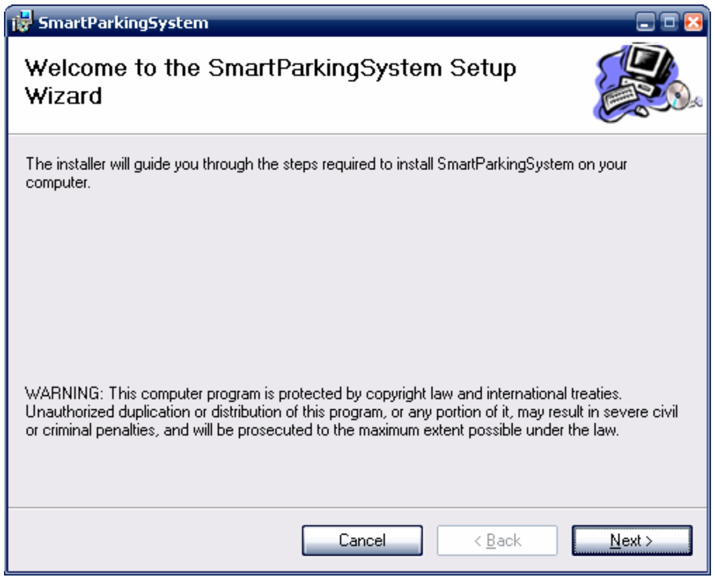


Figure 3.23: Windows Installer



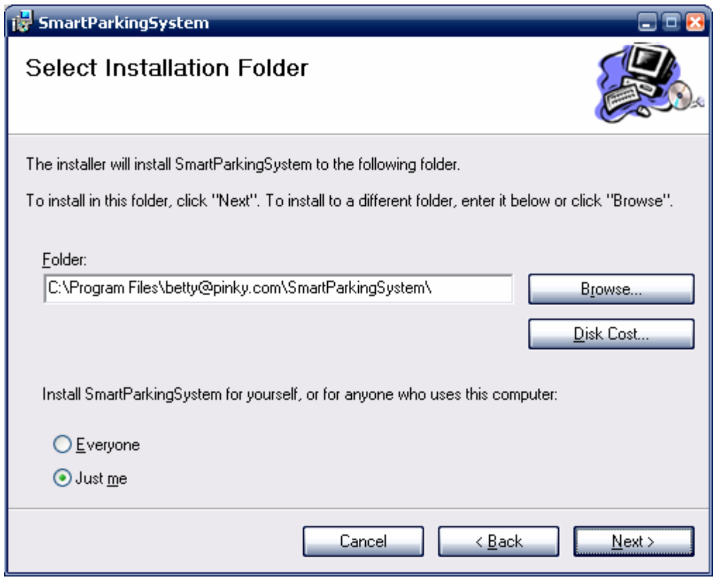
Welcome to the SmartParkingSystem Setup Wizard

The installer will guide you through the steps required to install SmartParkingSystem on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

Figure 3.24: SmartParkingSystem Setup Wizard

Click on the Next button to continue.


SmartParkingSystem

Select Installation Folder

The installer will install SmartParkingSystem to the following folder.

To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

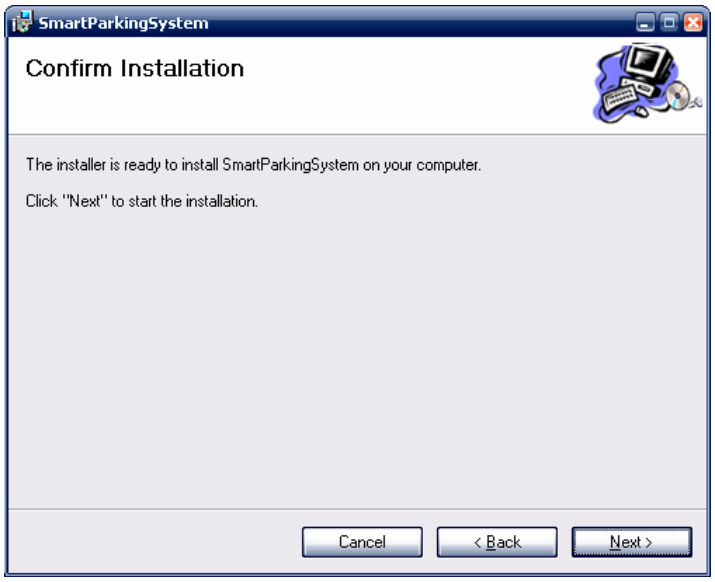
Folder:
C:\Program Files\betty@pinky.com\SmartParkingSystem\

Install SmartParkingSystem for yourself, or for anyone who uses this computer:

○ Everyone
[✓ Just me]

Figure 3.25: Select Installation Folder

Click next and follow the Screens to install the application. Once the installation is complete, make sure that a shortcut is created on the desktop and that it correctly launches my SmartParkingSystem application.

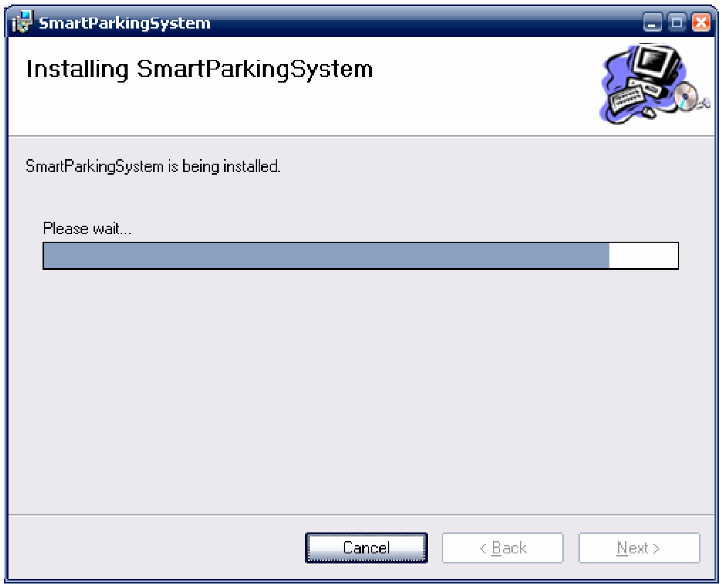

SmartParkingSystem

Confirm Installation

The installer is ready to install SmartParkingSystem on your computer.

Click "Next" to start the installation.

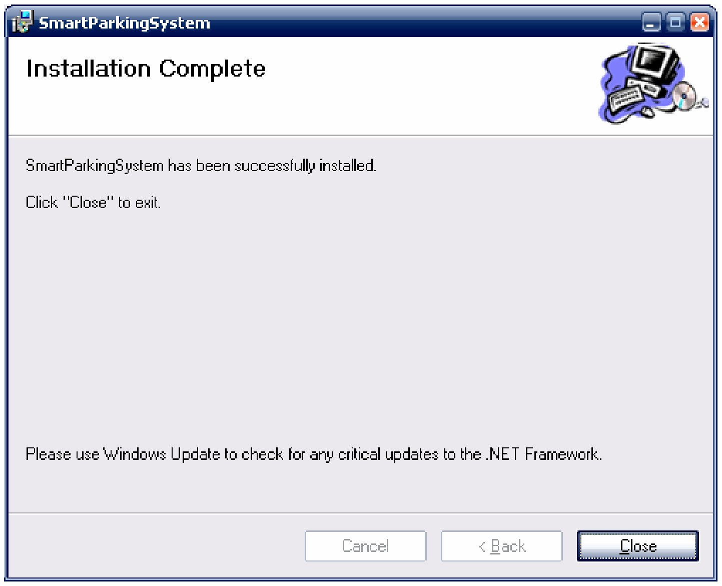
Figure 3.26: Confirm Installation



SmartParkingSystem is being installed.

Please wait...

Figure 3.27: Installing SmartParkingSystem



SmartParkingSystem has been successfully installed.

Click "Close" to exit.

Please use Windows Update to check for any critical updates to the .NET Framework.

Figure 3.28: Installation Complete

3.6.4 Uninstalling the Application

Once I have successfully installed the application, I can also uninstall it by selecting Uninstall from the Project menu. I could also uninstall the application from the Add/Remove Programs list in control panel.

I can add as many Setup and Deployment projects in one solution as I want. Each of them can have different settings and options.

By adding a few bits and pieces to my project, I saved my selves from writing code to install our application - not to mention uninstalling the same application. [6]