Stanley Steemer Job, Customer, and Inventory Tracking Application

By

Ralph F. Mitchell III

Submitted to the Faculty of the
Information Engineering Technology Program
in Partial Fulfillment of the Requirements for
the Degree of Bachelor of Science
in Information Engineering Technology

University of Cincinnati
College of Applied Science

March 2005

Stanley Steemer Job, Customer, and Inventory Tracking Application

by

Ralph F. Mitchell III

Submitted to
the Faculty of the Information Engineering Technology Program
in Partial Fulfillment of the Requirements
for
the Degree of Bachelor of Science
in Information Engineering Technology

Ralph F. Mitchell III — Date: 3/7/05

Professor Russell McMahon, Faculty Advisor — Date: 3/7/05

James F. Sullivan, Department Head — Date: 7 March, 2005

Acknowledgements

I would like to thank Dave Tarter, manager of Stanley Steemer of Northern Kentucky, for allowing me to model my application after his company and providing information whenever I requested it. I would also like to thank professors Russ McMahon, Robert Schlemmer, John Nyland, and Anu Prahbakar for support and suggestions during the development of the program. Thank you to everyone that helped not mentioned above for all the help provided to me when I needed it.

Table of Contents

List of Figures

Abstract

Stanley Steemer Job, Customer, and Inventory Tracking Application is a dynamic C# project designed to expedite information entry and retrieval as well as eliminate paper documentation. The current information resides on paper only, creating havoc with information retrieval and business forecasting. Through this application, the information is made readily available for entry and retrieval, allowing information analysis covering long periods of time. The application includes various reports to display the information in a logical manner. The information is provided by using both established technologies, Microsoft SQL Server 2000, and new development technologies, C# programming language.

1. Description and Intended Use

1.1 Statement of the Problem

Stanley Steemer of Northern Kentucky is a franchise of Stanley Steemer International Inc. In the daily running of the company, the manager or secretary will record jobs from both current and new customers. The manager is also in charge of maintaining the condition of the trucks, monitoring inventory, and tracking other aspects of the company. Currently, everything is tracked via a pen and paper system, this can be tedious, time-consuming, and inefficient. As the company continues to grow, the ability for the company to adjust and maintain order using their current tracking system will become impossible.

1.2 Solution of Problem

The application is created using the C# programming language. Combining the C# language with a SQL Server database for information storage will provide a robust program while maintaining an easy-to-use interface.

The application satisfies the needs of the company. The main areas that I focused on are digitizing paperwork, allowing for easier tracking of customers, supplies, and employees, and producing various reports. A database stores the information entered into the program. The application includes many features customized specifically for the franchise. The application allows input of past jobs, as well as current and future jobs. The features of the application include employee tracking by tracking which job the employee is assigned, how much time the employee is taking per job, and what items the employee is selling at the job site. Reports can be created that summarize tracking of the

Daily/Weekly/Monthly sales information per employee; tracking information to provide a quantitative and qualitative analysis of the employee's job performance; and tracking sales that allow management to improve selling and cleaning techniques. It will provide inventory tracking by allowing a daily inventory; track any missing or lost items, to prevent company loss; it will track items sold, to see if more of particular item is required on a daily basis. The application allows for in-shop inventory by tracking what items are being sold and in what amounts and facilitating monitoring of items sold over a period of time. Another feature is providing managerial reports such as tracking quality of work based on reports filed by customer call-ins requesting work to be redone or property damaged. It tracks the inventory of trucks and shop to allow ordering of supplies to be more proactive, employee's performance to determine advancement and quarterly and yearly reports tracking any information regarding employees. It also provides tracking of individual jobs, allowing easy qualitative and quantitative data to be extracted for reporting and allows research to determine time management principles.

2. User Profiles

2.1 Manager

The manager has little experience with the program interface though they have much experience with the paperwork for which the application is based. They also have moderate experience in Windows as and end user. They understand the menu bar, text box, buttons that are present in many programs. There is a great understanding of the paperwork and tasks that are required to run the company.

The program will need to follow the logical entry of data based on the paperwork, maintain an easy to use interface, and include all information present on the paperwork. There will be certain features available only to the manager.

2.2 Non-Manager

The non-manager has little experience with the program interface though she has moderate experience with the paperwork for which the application is based. They also have moderate experience in Windows as an end user. They understand the concepts of the menu bar, text box, buttons that are present in many programs. There is a fair understanding of the paperwork and tasks that are required to enter information about jobs.

The program will need to follow the logical entry of data based on the paperwork, maintain an easy to use interface, and include all information present on the paperwork. The non-manager will have a limited access to the program.

2.3 Application Designer

As the designer, I have experience in the interface design and with the application. I have extensive experience with Windows OS and applications within the OS. I have little experience running any aspect of the company, but have become familiar with the paperwork.

I have full access to the application for design and testing. After completion I will only need access to upgrade or change any aspects deemed necessary by management for both the database and application.

3. Project Design Protocols

3.1 Senior Design Requirements - Deliverables

The deliverables for the project will include:

- A Windows Job and Inventory Database Application.
- A back-end database that will store information entered by the application. It will be developed on SQL 2000.
- A dynamic interface developed in C#.
- Multiple-user authentication through SQL for security.
- An easy to follow, logical transition for entering information.
- Unique forms for entry of information requested by the client.
- Ability for authenticated users to (based on rights granted)
  - Add customers, jobs, and employees
  - Add new products and inventory items
  - Create and print reports of varying types bases on rights
  - Find information to aid them in the application use in the help file

3.1.1 Application Programming

The application developed collects data used everyday by the company. It manipulates information for entry into a storage database as well as for printouts.

3.1.2 Database Management

A database holds all the information entered and retrieved by the application. It is on a server running SQL Server Enterprise edition.

4. Design Details

4.1 Timeline

Task	Date Started		Date Finished	
Task Research	Wed	1/7/04	Wed	3/10/04
Design Database Structure	Mon	1/19/04	Fri	4/30/04
Write Proposal	Mon	2/16/04	Wed	3/10/04
Senior Design I Presentation - Proposal	**Wednesday, March 10, 2004**			
Design Program GUI	Mon	2/16/04	Fri	4/30/04
Proposal Rough Draft Due	Wed	2/18/04	Wed	2/18/04
Finalize Proposal/Create Presentation	Wed	2/25/04	Wed	3/10/04
Write C# Code	Wed	3/10/04	Tue	6/22/04
Create Database Stored Procedure Code	Mon	5/3/04	Fri	8/13/04
Review/Reanalyze/Redesign GUI	Mon	5/17/04	Fri	6/25/04
Reanalyze/Redesign/Rewrite C# Code	Wed	6/23/04	Tue	10/5/04
Senior Design II Design Freeze Draft	Mon	8/2/04	Mon	8/2/04
Senior Design II Presentation – Prototype	**August 23, 2004**			
Test C# code/stored Procedures/DB Design	Wed	10/6/04	Tue	1/25/05
Write Manual/Final report/Etc	Tue	1/1/05	Fri	3/4/05
Finalize Application	Wed	2/9/05	Tue	2/15/05
Senior Design III Presentation – Final	**March 7 – 13, 2004**			

4.2 Budget

Item	Cost	Source of Cost
Microsoft SQL Server 2000 Standard	$5,000.00	Microsoft SQL Server
Windows 2003 Server and .NET	2,500.00	Hardware, Windows license
Windows XP Station with .NET	500.00	Hardware, Windows license
TOTAL	**$8,0 00.00**	

4.3 Software Requirements

The software requirements for the application are a Windows Server operating system with the .NET Framework installed. I chose the Windows 2003 Server Standard edition for purposes of the installation. Microsoft SQL Server 2000 is to be installed on the server.

Each machine connecting to the server requires a Windows operating system that will handle the .NET Framework installation.

4.4 Hardware Requirements

The hardware requirements to run the program are a server that can handle the Windows server operating system as well as the installation and running of Microsoft SQL 2000 Server Standard edition. The more memory the system can handle, the better for performance. It will need to contain a network interface card that is capable of connecting to the internal network.

The client desktops are similar in they need to meet the requirements to install the Windows operating system as well as the .NET Framework. Again the more memory the system has, the higher the performance. It will also require a network card that can connect to the internal network.

5. Proof of Design

5.1 Application Forms

Screenshots of the application provided below with a description of what each facet of the program entails.

5.1.1 Application Startup

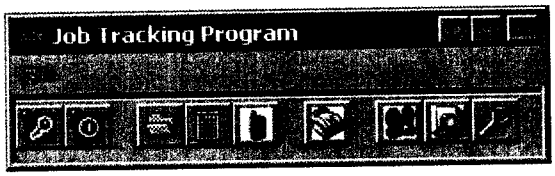

Figure 1

The application starts up with the screen above. It is a basic link to each of the prominent forms. Shown is a menu that is login specific, giving the manager the option to edit employees, run reports, and run the administration features. Everyone that logs in will have the ability to login as a different user, logout, access the job form, job

assignment form, employee-call in form, and call-back form. Figure 2 is an example of a non-manager login.

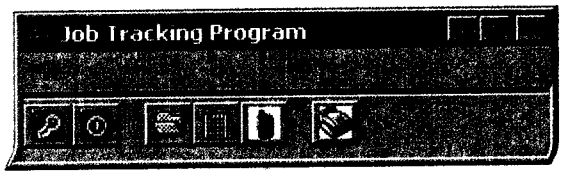

Figure 2

5.1.2 Customer Form

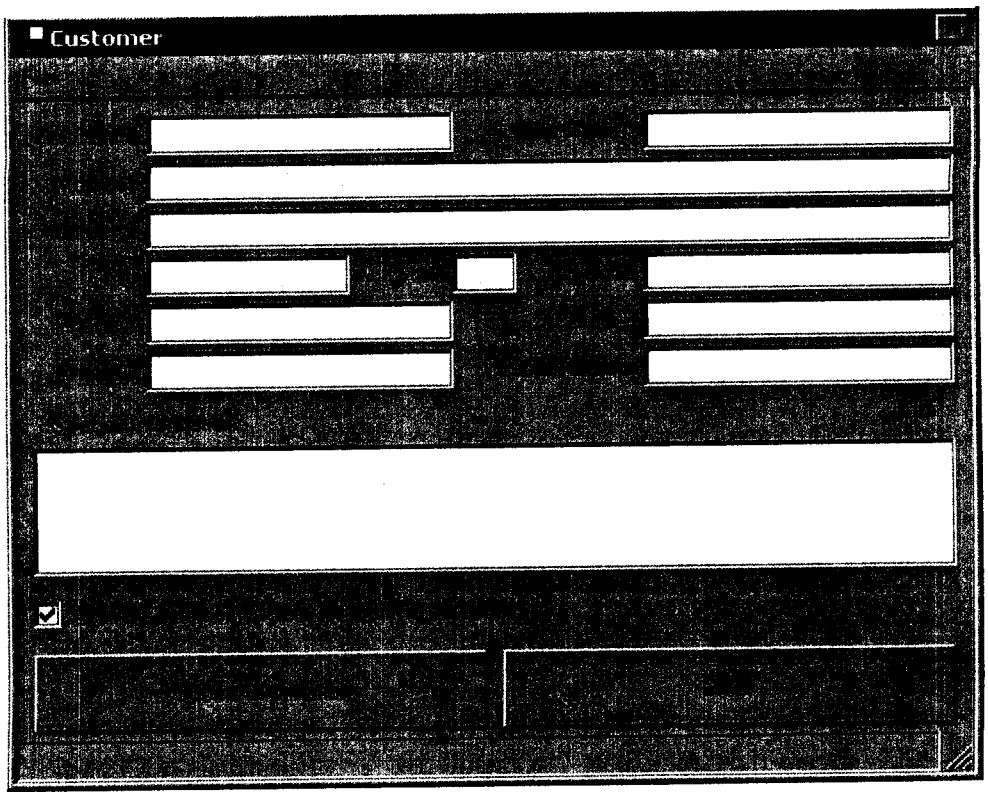

Figure 3

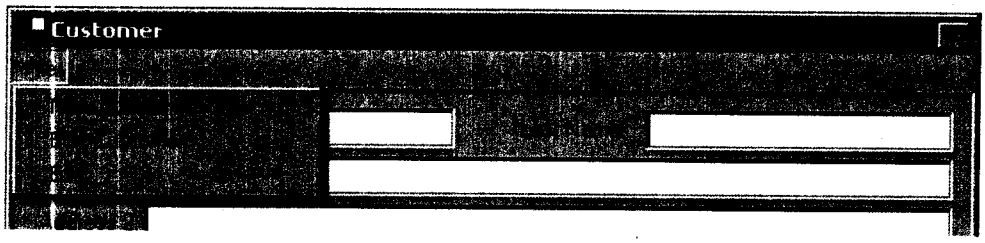

Figure 4

On this form, the customers' details are displayed. The Enter Customer button is the master control for this, allowing updates or new entries into the database. The file menu will allow a new customer to be entered, or to load another form, that displays the customer name, address, and phone numbers. The form that is loaded then passes the value back to the customer form where it can be edited and then updated. It is closed either by the Close button or the file menu choice Close, which contains the shortcut control Alt + F4.

5.1.3 Employee Form

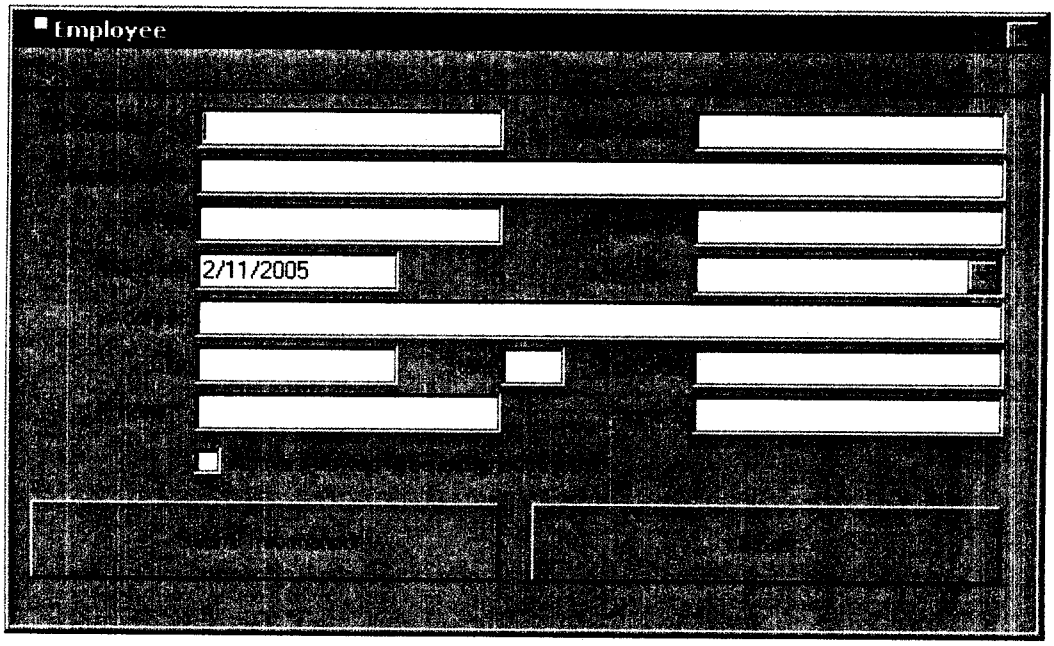

Figure 5

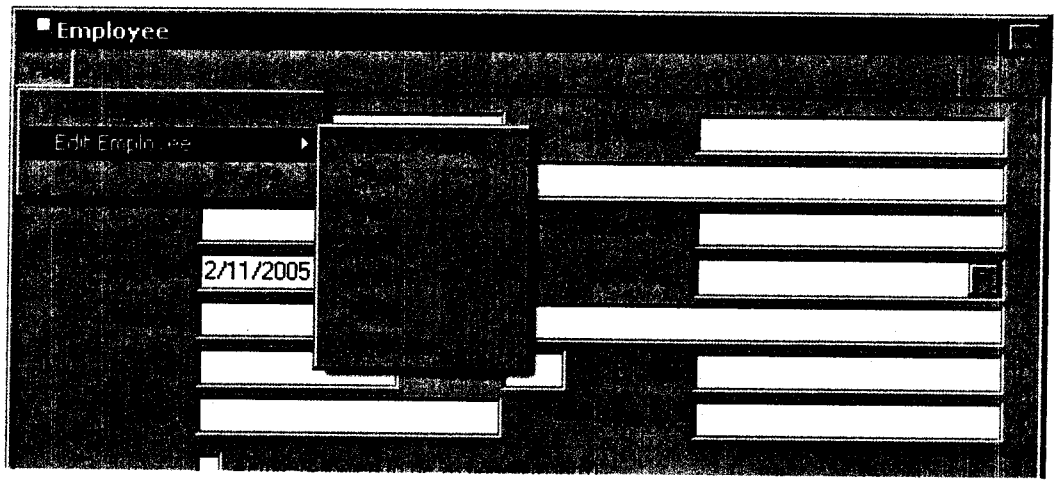

Figure 6

The Employee Form is similar in setup to the Customer form, with the exception of on load, under File -> Edit Employee, the list of employees is dynamically loaded into the menu (Figure 6). Employees are updated and added via the form. At the finish of this *form, an employee is created that can be used for jobs and login onto this application.*

5.1.4 Job Form

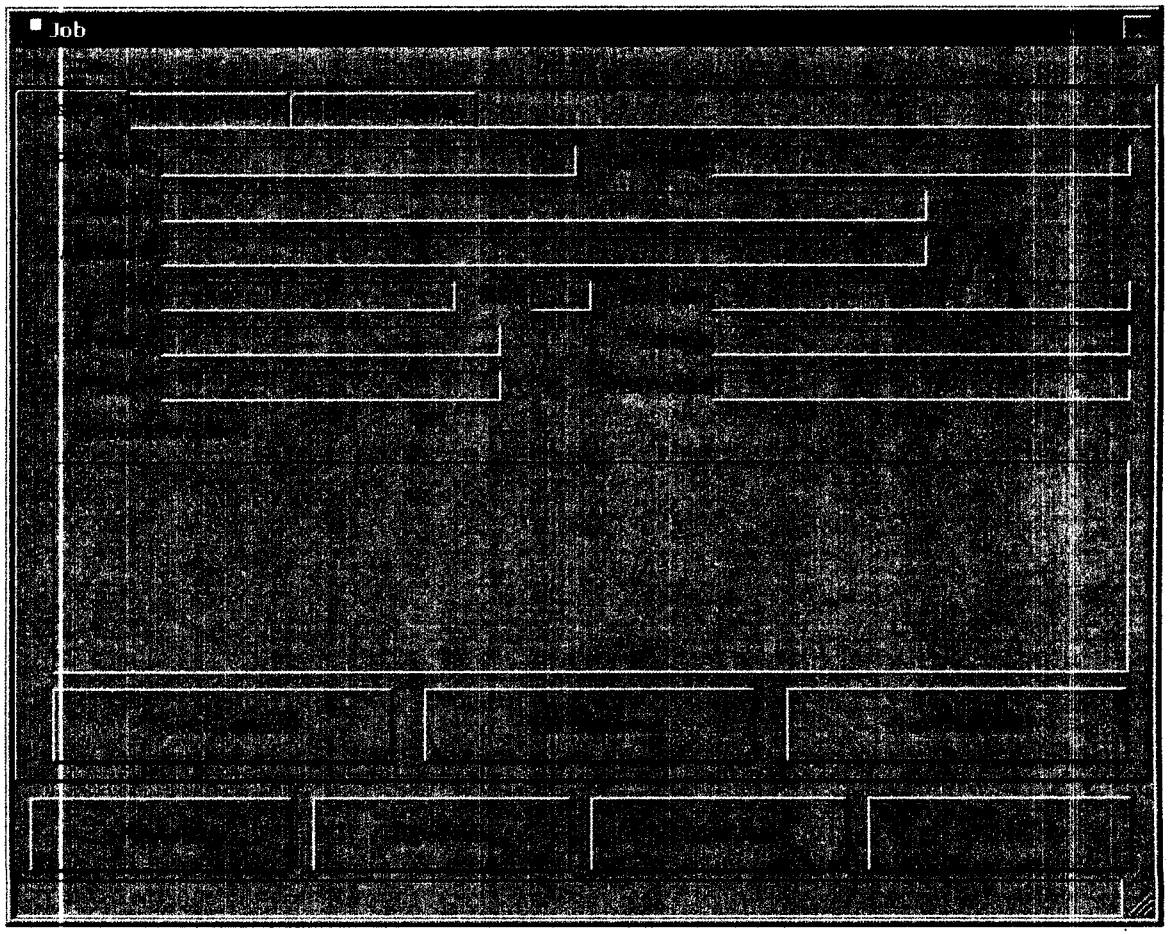

Figure 7

Here is the startup view of the form. It has a tab format, with Customers, Job Information, and Job Item Summary. The starting tab is the customers tab, where you can select a former customer, edit the customer selected, and/or enter a new customer. The New Job, Load Job, Submit Job, and Close buttons are available from anywhere on the form. The menu does similar formats of new jobs, load jobs, and close, as well as the Tab menu that will toggle between tabs.

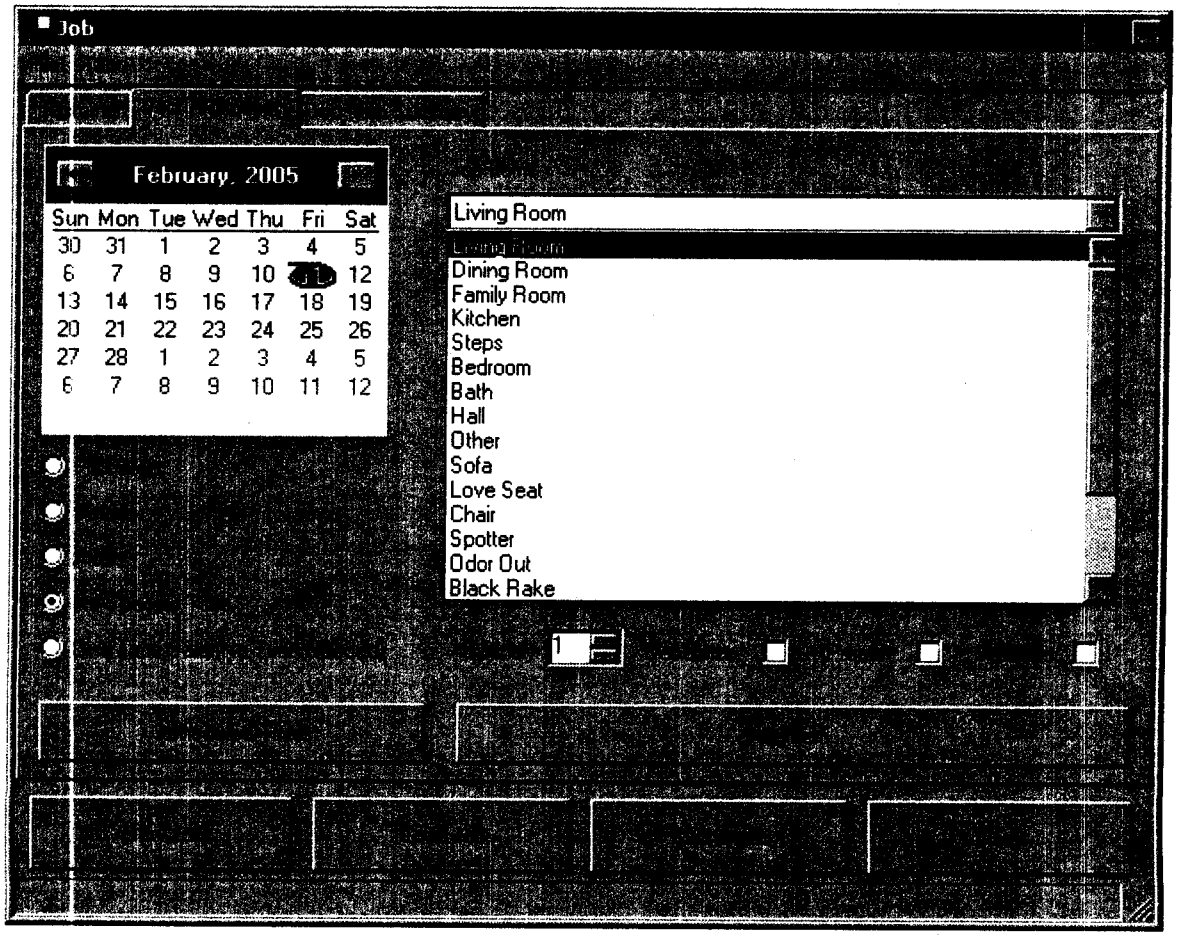

Figure 8

The Job Information tab allows for a date selection, specifying a time selection, adding items to the job, and opt which details (deodorized, protected, and details) are assigned to each item. The drop down box is read-only but contain the ability populate from database table what job items are there, thus allowing the business to expand job items easily.

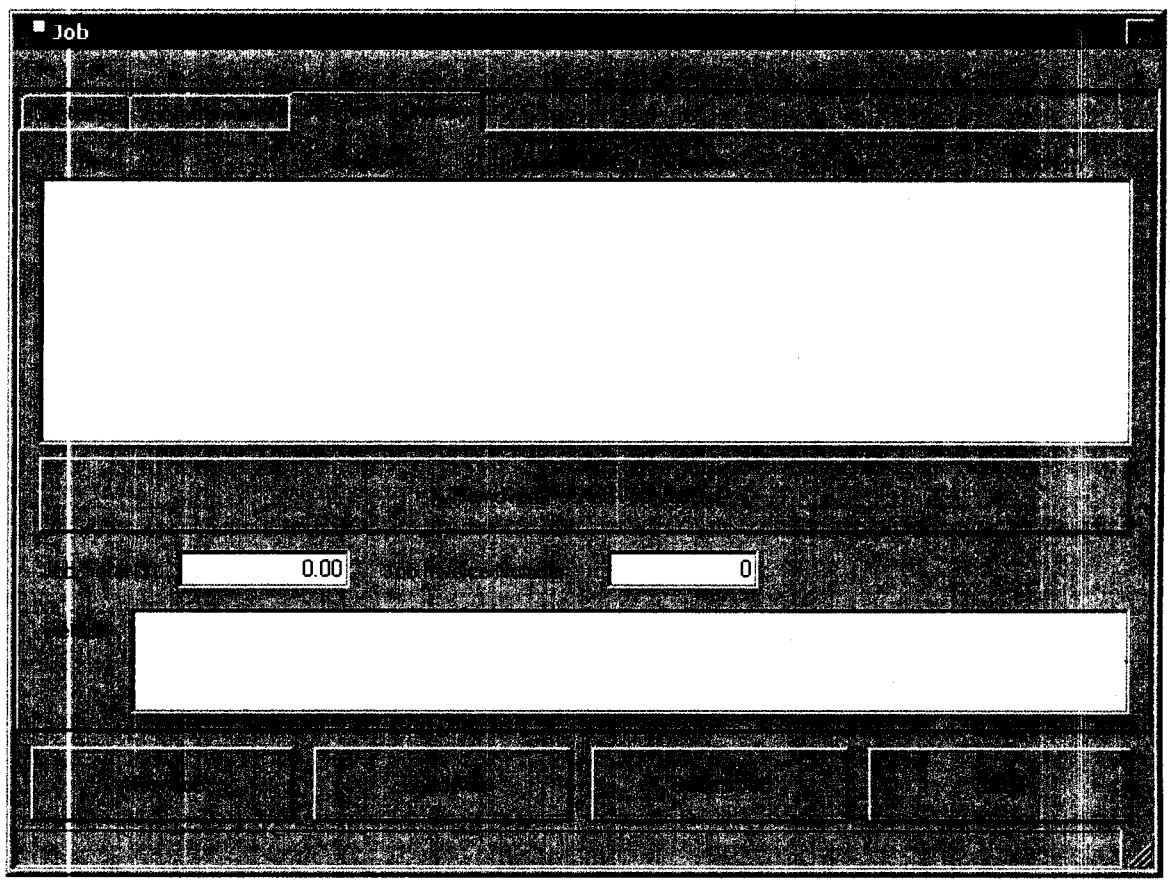

Figure 9

The Job Item Summary tab shows what job items have been added to the job, also allowing for items to be removed. It is a check box, so that multiple specific items can be removed. It also allows for the detailed list to be used to specify a specific price in the Job total.

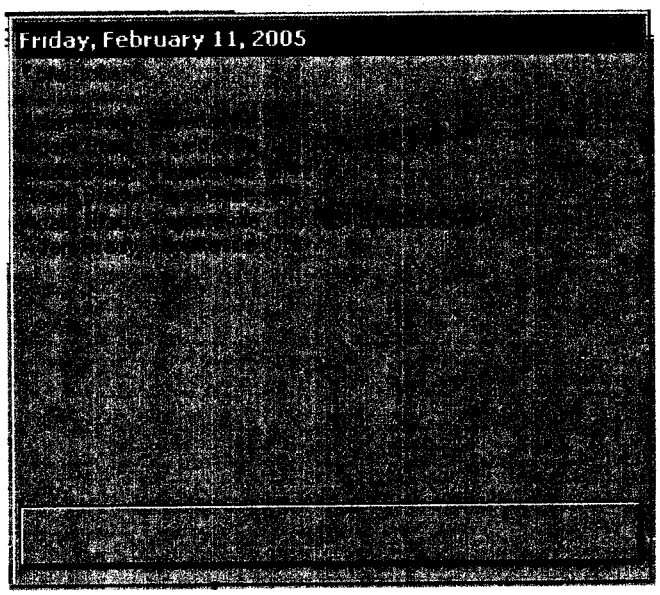

Figure 10

The scheduled Times button will display a list of jobs scheduled for the date selected on the Job Information tab. It will show upon being clicked and OK will take you back to the previous form.

5.1.5 Job Assignment Form

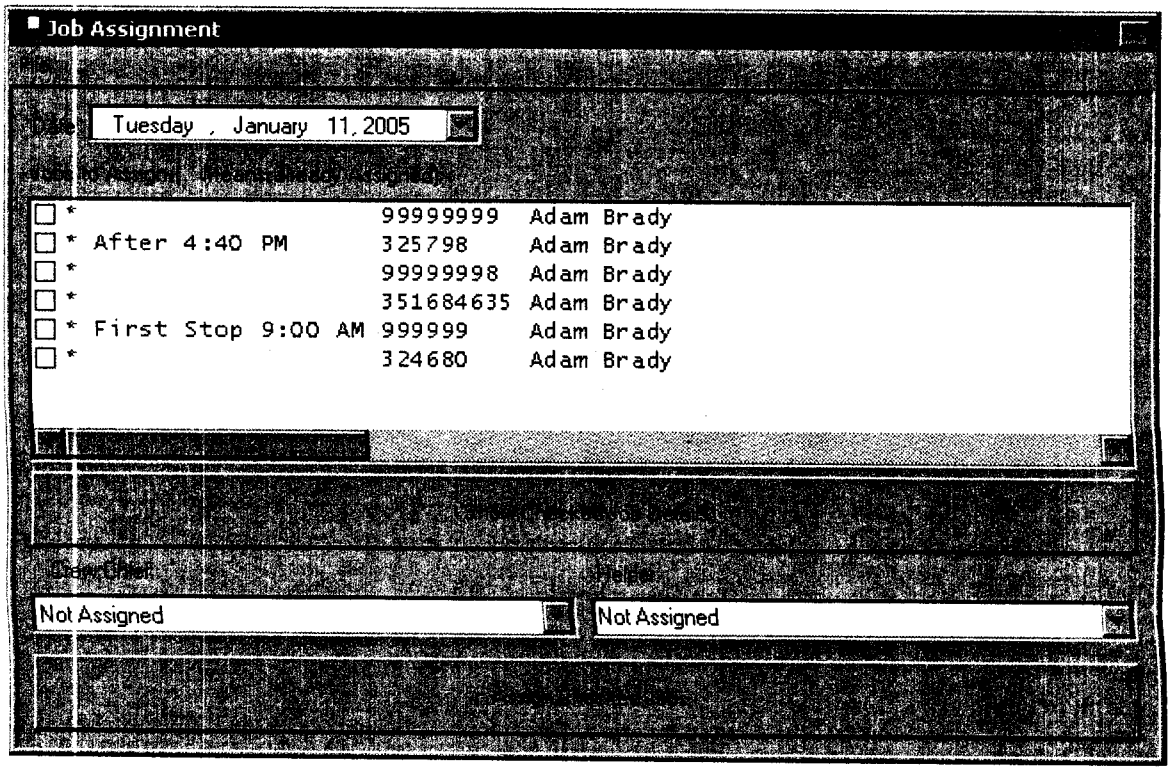

Figure 11

This form displays job information for jobs displayed (Figure 11). This shows the specific time requirements, job invoice, and customer. This form allows bulk assignment of the crew chief and helpers to the job by checking the jobs, selecting a crew chief and helper and clicking the Assign Checked Jobs button. An asterisk will appear by jobs that have crew assigned. Also, a person can view details about the jobs by checking the jobs and clicking Show Checked Job Details. This will be reviewed next.

5.1.6 Selected Jobs Details Form

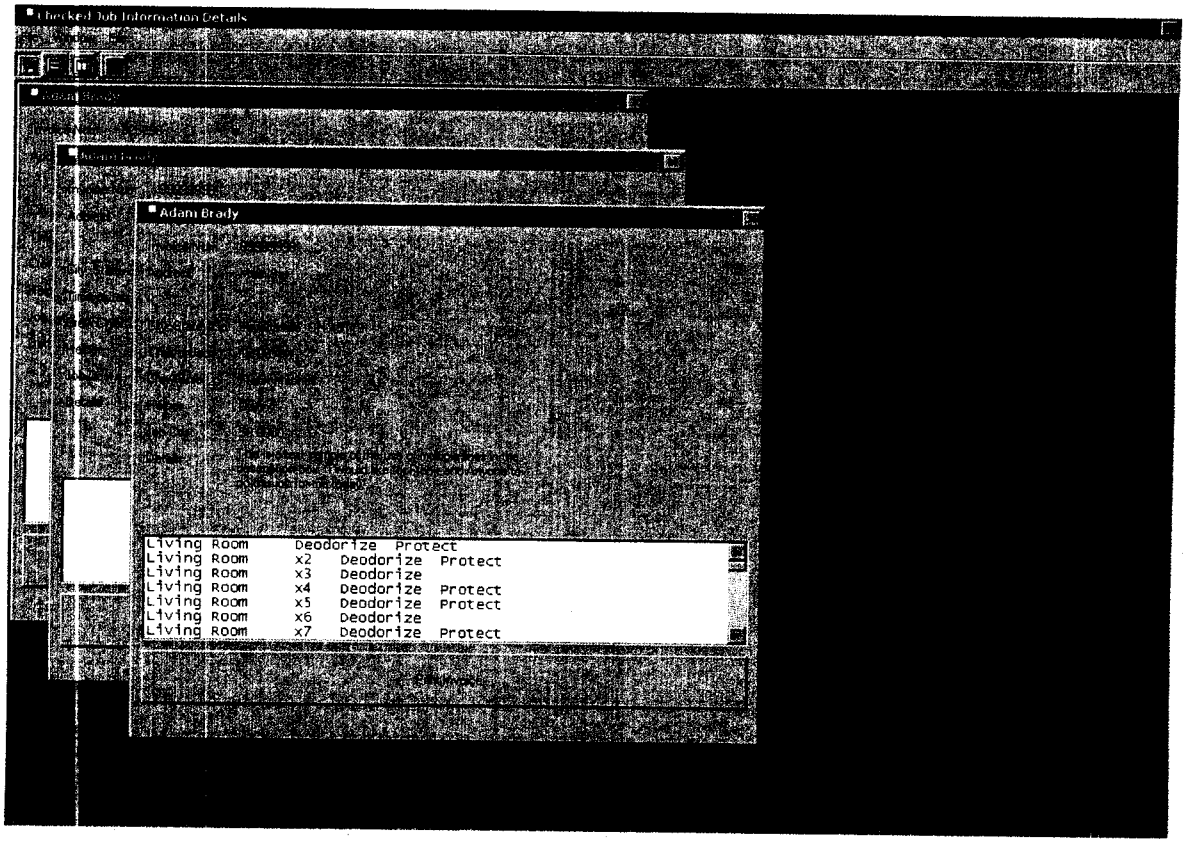

Figure 12

This form displays the detailed information about jobs. Each job can be moved or arranged in standard windows positions (tile horizontally, tile vertically, and cascade). It also allows for a single job invoice to be printed onto an invoice sheet (Figure 13).

5.1.7 Employee Call-In Form

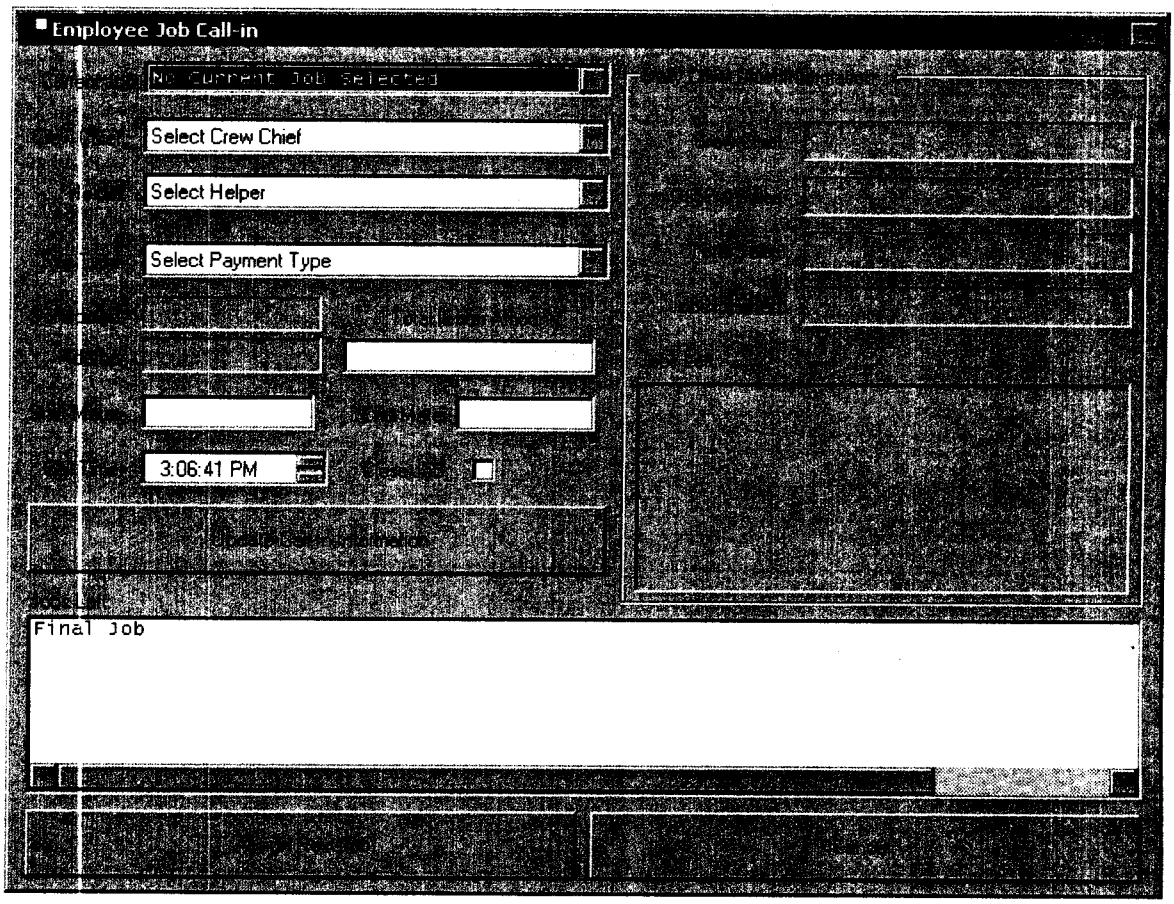

Figure 13

This form is for input of information after the job has been completed. It shows only today's jobs as to eliminate confusion. It will auto-populate the information if it exists. This form will tally an employee's daily work load on the right, as to allow the manager to distribute jobs accordingly. It also allows the assignment or re-assignment of jobs to the crew. This is to prevent going between multiple forms to change who the job is assigned to.

5.1.8 Manager Call-back Form

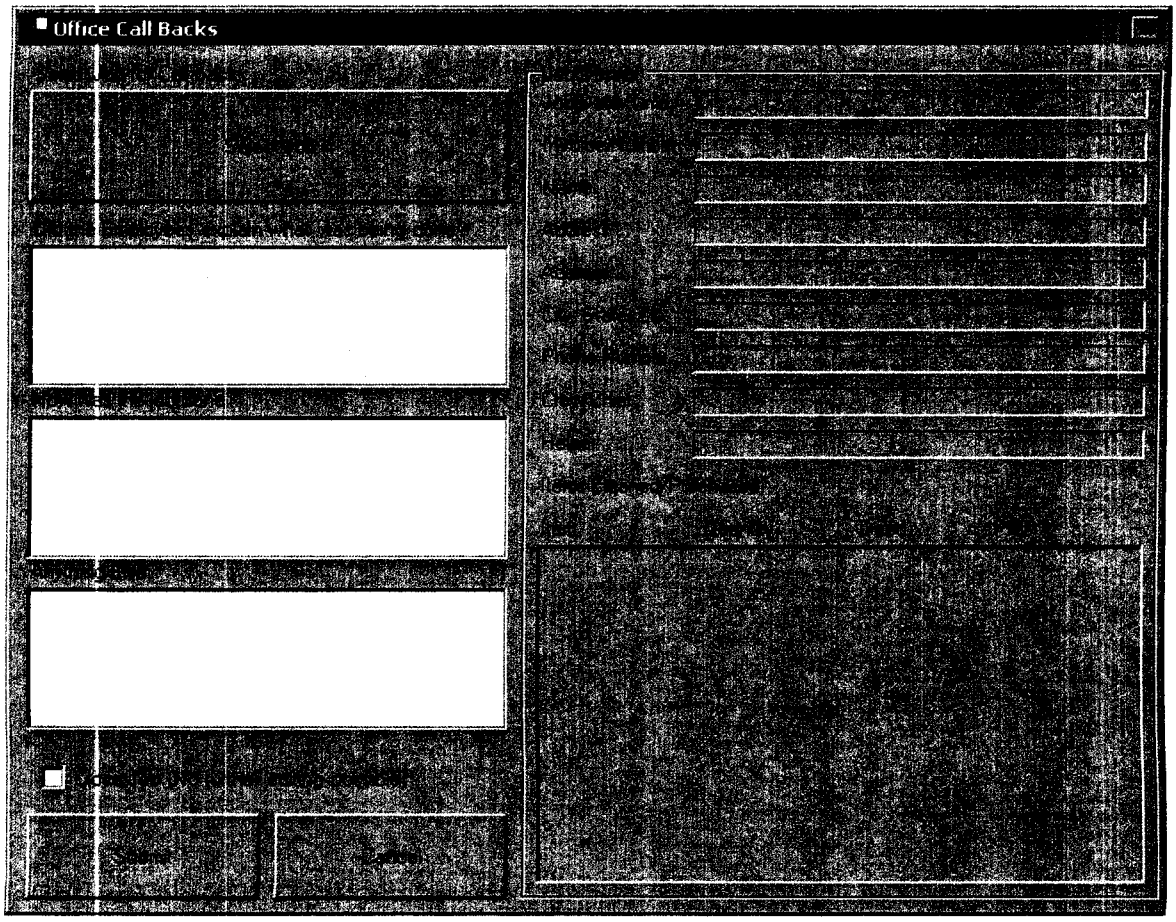

Figure 14

This form is for documenting the information when a manager checks the quality of work by contacting the customer. The three questions listed about are asked. By clicking on select jobs, the list is populated with jobs not closed yet. This is not date specific. Information about the job is loaded on the right for review.

5.1.9 Reports Form

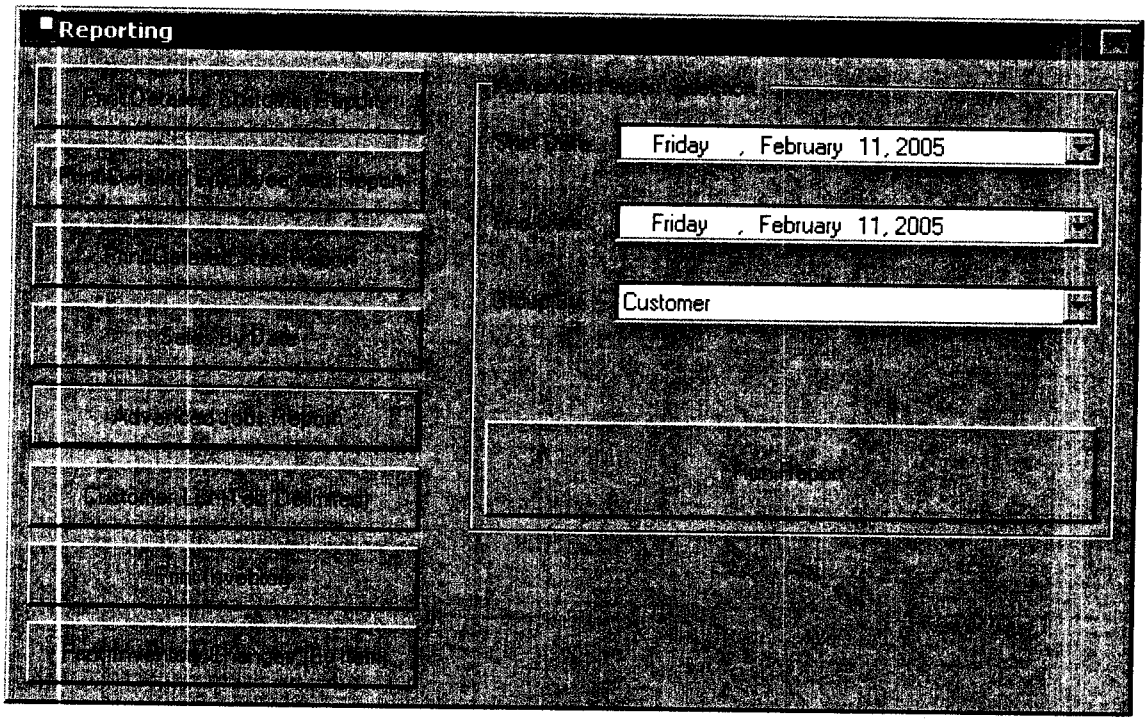

Figure 15

This form is for creating reports on paper or digital format. The detailed customer report will display the customer information and detailed information about all jobs that have been done for the customer. The Detailed Employee Report is similar, detailing jobs done by the customer. A detailed jobs report is similar, for one job only. Sales by Date will allow a report to be generated detailing sales by day, month, or year over a user selected time period. An advanced job report will display information over a user selected period of time and display general information about jobs; it is customizable by selecting whether to sort via customer, crew chief, or date. The Customer list will create a tab-delimited file containing customer information for use in creating labels. It was not directly outputted to file because different labeling setups might require different print setups, but most allow population via tab-delimited files. The print inventory will print a

current list of inventory in stock. The Print Inventory Changes will print, given a time frame, the changes made to the inventory, so charts, diagrams, or reports can be generated. It comes in both tab-delimited format and printouts.

The printing will allow a system printer (one seen by the OS), to be dictated as the output location, if tab-delimited file option is not chosen. This allows for paper output or printing to any other system created printers (Adobe Acrobat PDF creator, Microsoft Office Document Image Writer, etc). Example reports are located in Appendix A.

5.1.10 Administration Form

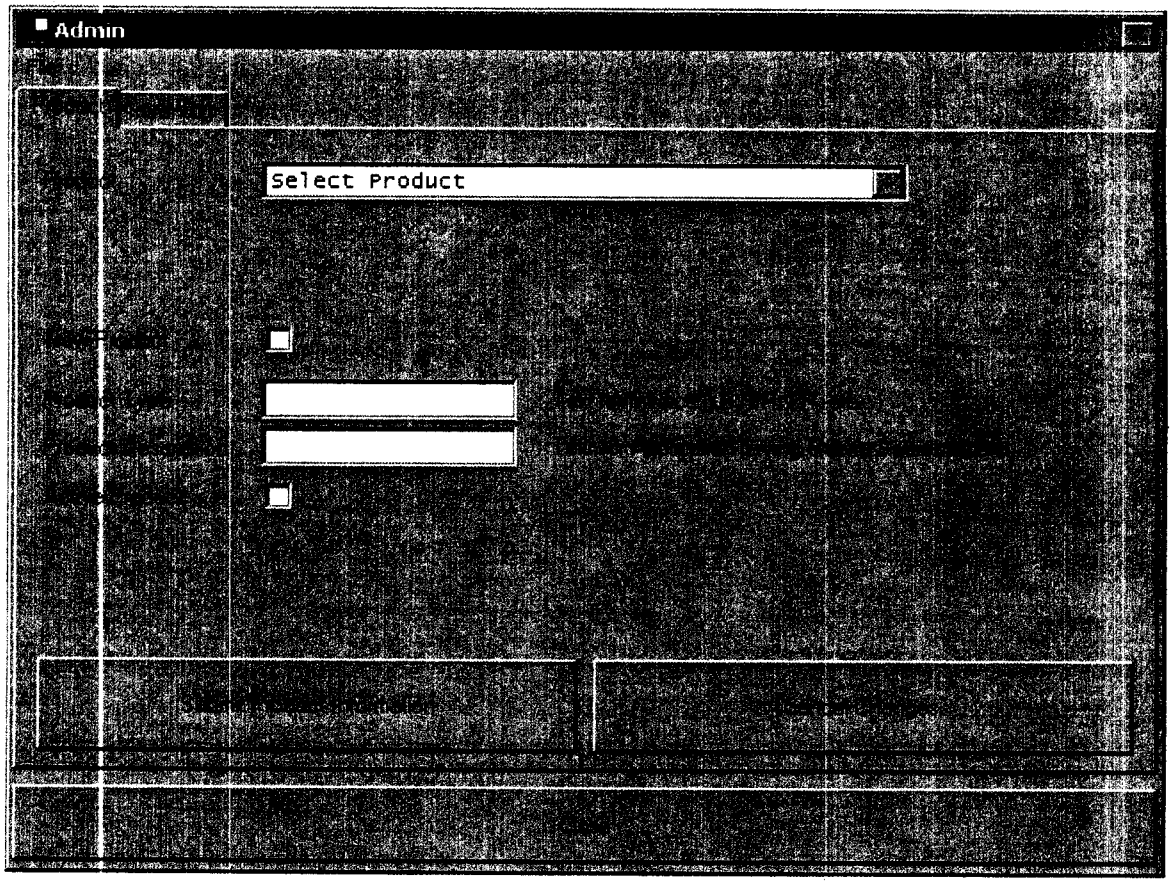

Figure 16

The administration form allows for information to be entered such as products that can be assigned to jobs. It controls how items are available by making them active or not. It

also does the same for the inventory, allowing changes in the inventory to be recorded in the database.

5.1.11 Job Select Form

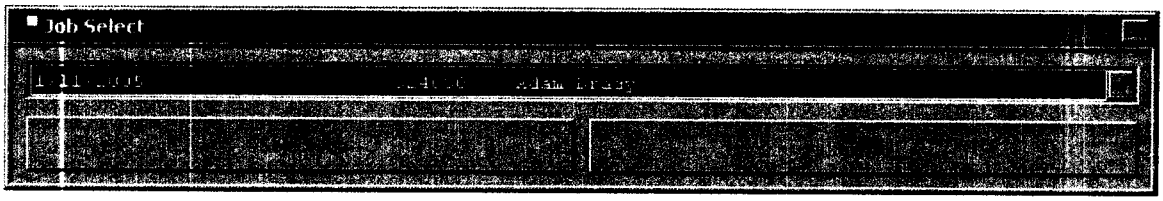

Figure 17

This form has the simple duty of displaying jobs for the person to select. Once this form has been called, it auto-populates with the jobs from the database.

5.1.12 Customer Select Form

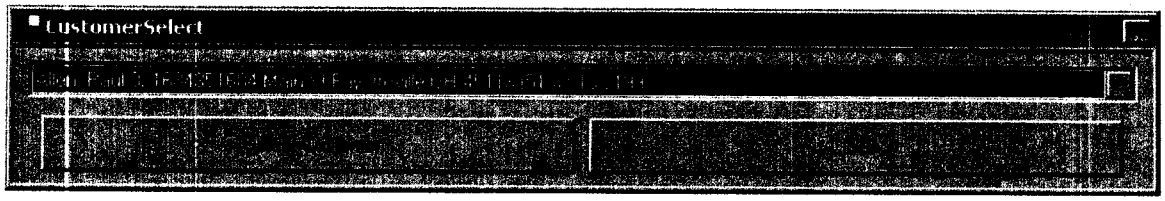

Figure 18

Similar to the form above, it auto-populates with customers for which to select.

5.1.13 Employee Select Form

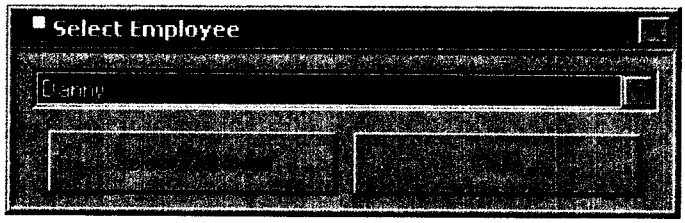

Figure 19

As same as the previous two forms, it auto-populates with employee for which to select.

5.1.14 Login Form

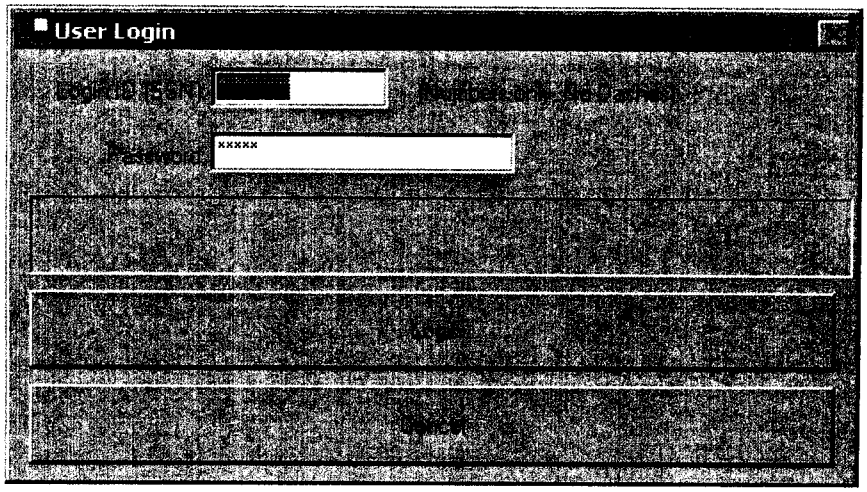

Figure 20

This form is one of the first seen upon entering the program, it allows the program user to enter information to grant access to the program. If the user cancels the login, the application shuts down to prevent unauthorized access.

5.1.15 System Tray Icon

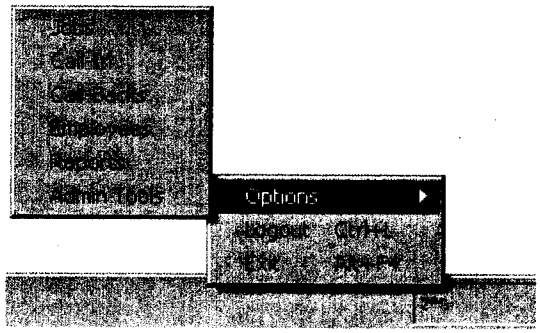

Figure 21

This system tray icon allows quick access to many features of the program. It is logon dependent as to which options it shows. It works from the main menu, and will not transfer forms if another is selected.

5.2 Application Layout

5.2.1 Application Diagram

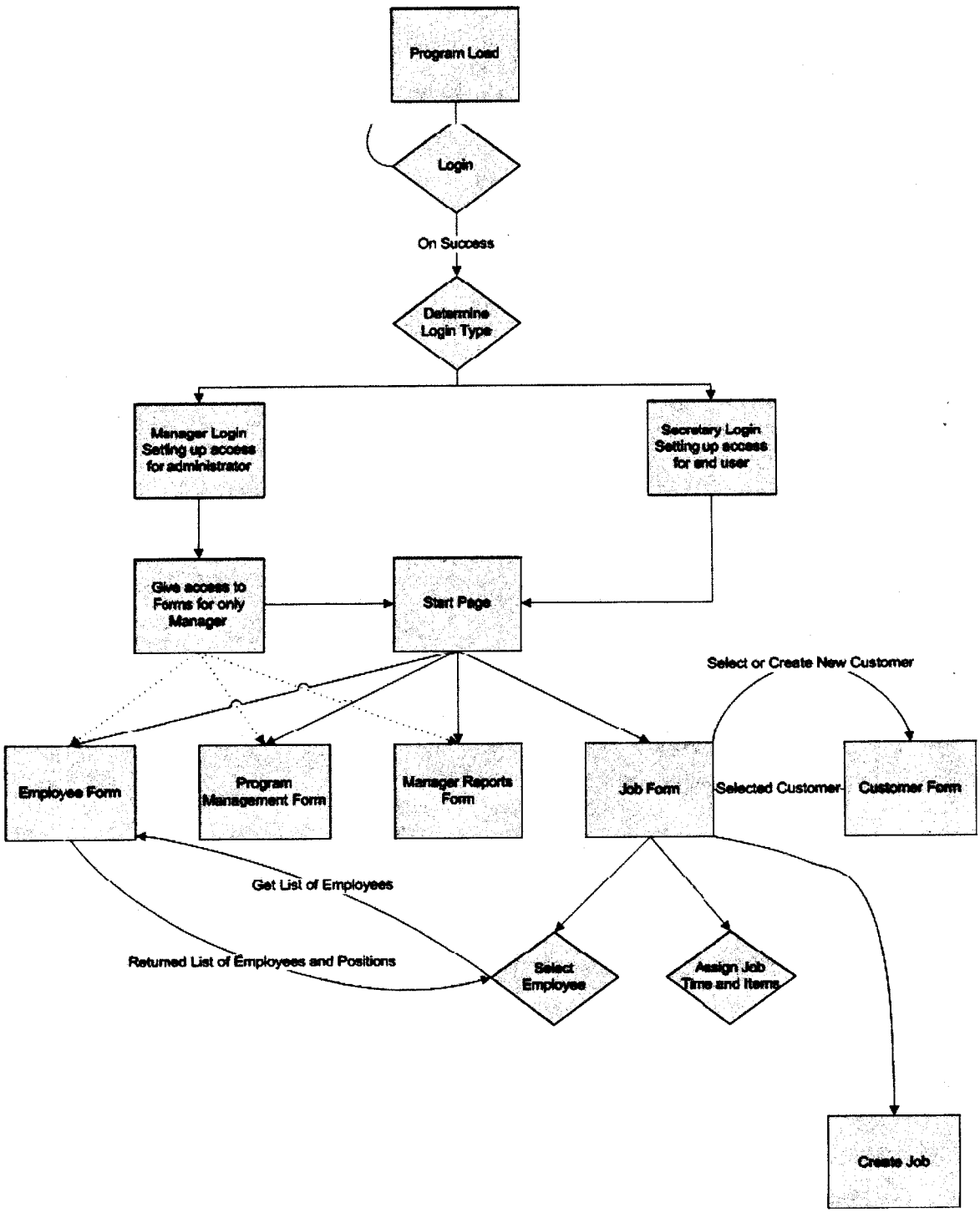

Figure 22

Figure 20 displays the flow of the program upon startup. There are going to be two types of access, managerial and non-manager roles that are accessible via the login. Certain privileges to forms are given to the manager. They are shown via the dotted lines in Figure 20. This is how I propose to place security, by using a login form that checks against the database table employees for correct login names and passwords.

This is an overview of the basic application flow. Below are more detailed actions on each of the forms.

5.2.2 Database Layout

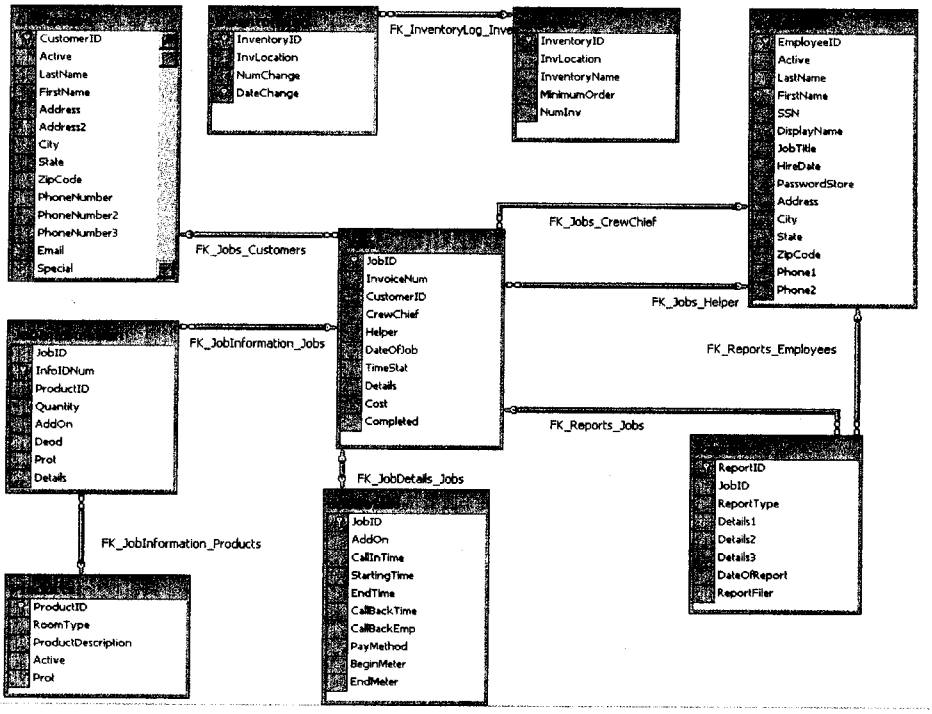

Figure 23

Figure 21 shows the layout of the database from which the application will draw its information.

5.2.3 Other Design Protocols

As a Windows application, the color scheme will use the default operating system color scheme. There are areas where I have dictated what font will be used, to allow a fixed width font to be used for easy formatting and following of text. There will be no icons, and the only graphical representations will be that of the Title bar with it's minimize, maximize, and/or exit buttons.

The help will be a manual and text file accompanying the program. It will be a basic overview of what was said above, the general layout of the items as well as what to do in order to use them.

5.3 Testing Procedures

The proposed testing will be done both in programming and reviewing stages. The programming testing was done during programming to test efficiency and stability of code. Error checking for run-time, compile, and logic errors were reviewed. At the end of coding, a vigorous test was done to check the handling of the program as well as the database, this focused on run-time and logic errors of the program..

6. Conclusions and References

6.1 Conclusions

This application was created in response to a need for digitizing information for Stanley Steemer of Northern Kentucky. I created a dynamic C# application that inputs information into a SQL Server 2000 database. It takes information into the system in a business specific logical order. An estimated budget of approximately $7,500.00 would be needed to take this program into production, where half of the cost is related to system

hardware and its software, and the other to the SQL licensing. Two levels of testing were performed to insure application stability.

6.2 Recommendations

While working on the project various problems were encountered. The programming language did not interface with outside applications as well as desired or documented. I also encountered several problems when trying to create reports. Another problem that I encountered was that of modifying the project to become a multithreaded application. The timing of the software gave much trouble, several times having to reanalyze the information to create an application that could handle the multiple threads. The last problem would be that of financial burden.

The reports problem was overcome after several attempts to use various programs (such as Microsoft Access) to generate and print the reports, eventually I created my own printouts.

The multithreaded issue I ran into took considerable time to overcome. I would probably have looked into making the application multithreaded at the beginning. Upon that requirement, the program would have been changed into multiple applications, one being the server-side application and the other to be a client-side application.

Finally, the financial burden of the application comes from using Microsoft SQL Server as my data storage. While the approach was feasible as a design project, I would probably change the design to work against an MS Access database to reduce cost.

7. Appendices and References

7.1 Appendix A

Customer Report **2/28/2005**

Name:	McMahon, Russ	Phone Number:	513 555 5555
Address:	407 Admin	Phone Number 2:	
Address 2:	OCAS Admin Building	Phone Number 3:	
City, State, Zip:	Cincinnati, OH 45202	E-Mail:	
Special:			

Invoice Number:	1	Crew Chief:		Begin meter:
Date of Job:	2/28/2005	Helper:		End meter:
Cost:	125.0000	Payment Type:	NA	
Details:				

Product Description	Quantity	AddOn	Deodorized	Protected
Kitchen	1	☐	☐	☐
Living Room	1	☐	☐	☐
Dining Room	1	☐	☐	☐
Steps	1	☐	☐	☐
Family Room	1	☐	☑	☑

Invoice Number:	2	Crew Chief:		Begin meter:
Date of Job:	2/21/2005	Helper:		End meter:
Cost:	99.0000	Payment Type:	NA	
Details:				

Invoice Number:	4	Crew Chief:	Frank Mitchell	Begin meter: 1.2
Date of Job:	2/28/2005	Helper:	User A	End meter: 1.3
Cost:	99.0000	Payment Type:	CA	
Details:				
	Ott is an ID			

Invoice Number:	6	Crew Chief:		Begin meter:
Date of Job:	2/28/2005	Helper:		End meter:
Cost:	12.0000	Payment Type:	NA	
Details:				

Invoice Number:	7	Crew Chief:	Frank Mitchell	Begin meter: 1.3
Date of Job:	2/28/2005	Helper:	User A	End meter: 1.4
Cost:	99.0000	Payment Type:	CA	
Details:				

1

Figure 24

Inventory Changes Report 2/28/2005

Green Rake	Number In Stock	Changed Date
	10	2/6/2005 9:16:26 PM
	3	2/7/2005 9:35:54 PM
Brown Rake	Number In Stock	Changed Date
	87	2/7/2005 7:56:37 PM
Green Mat	Number In Stock	Changed Date
	13	2/4/2005 9:16:38 PM
	15	2/5/2005 9:10:05 PM

Figure 25

Sales Report 2/28/2005

Date	Total Jobs	Total Cost	Total AddOn
1/28/2005	0	0	0
1/29/2005	0	0	0
1/30/2005	0	0	0
1/31/2005	0	0	0
2/1/2005	0	0	0
2/2/2005	0	0	0
2/3/2005	0	0	0
2/4/2005	0	0	0
2/5/2005	0	0	0
2/6/2005	0	0	0
2/7/2005	0	0	0
2/8/2005	0	0	0
2/9/2005	0	0	0
2/10/2005	0	0	0
2/11/2005	0	0	0
2/12/2005	0	0	0
2/13/2005	0	0	0
2/14/2005	0	0	0
2/15/2005	0	0	0
2/16/2005	0	0	0
2/17/2005	0	0	0
2/18/2005	0	0	0
2/19/2005	0	0	0
2/20/2005	0	0	0
2/21/2005	1	99	0
2/22/2005	0	0	0
2/23/2005	0	0	0
2/24/2005	0	0	0
2/25/2005	0	0	0
2/26/2005	0	0	0
2/27/2005	0	0	0
2/28/2005	9	687	0

Figure 26

Inventory Report **2/28/2005**

Inventory Name	Number In Stock	Order When At
Green Rake	3	4
Brown Rake	87	4
Green Mat	13	5

Figure 27

7.2 Appendix B

Code Snippets

This is the code for the class called to run the timer thread:

```
private void dbTimerStart()
{
        System.Timers.Timer threadTimer = new System.Timers.Timer();
        threadTimer.Elapsed+=new ElapsedEventHandler(OnTimedEvent);
        // Set the Interval to 10 minutes.
        threadTimer.Interval = 600000;
        threadTimer.Enabled=true;
}
```

This is the code for the timer to call on the database update thread.

```
private void OnTimedEvent(object source, ElapsedEventArgs e)
{
        Thread dbUpdate;
        ThreadStart dbTStart = new ThreadStart(dbThreadStart);
        dbUpdate = new Thread(dbTStart);
        dbUpdate.Start();
}
```

This is the code to gather the rows between a date range, sorted from the dataset.

```
        string rowFilter = @"DateOfJob > '" + start + @"'";
        rowFilter += @" AND DateOfJob < '" + end + @"'";

        DataRow[] SortedJobs = sql.dsApp.Tables["Jobs"].Select(rowFilter, RowSort);
```