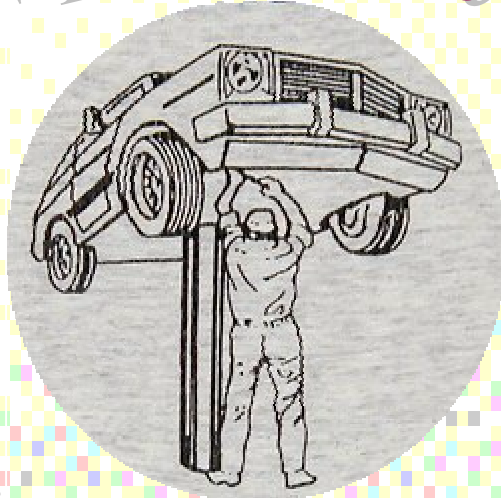


# Service Tracking System

Sponsored by



Heights Service Station

CIS 491-101  
Professor Eljabiri  
Fall 2002

Project P00046

Richard J. Sonderfan  
Marc DeFilippo  
Partha Majumdar  
Rima Patel  
Vidhu Sukheja  
Tony Yang

## Table of Contents

1. Introduction.....	4
1.1. Abstract.....	4
1.2. Team Members and Job Descriptions.....	5
1.3. Background.....	6
1.4. Problem Statement.....	8
1.5. Previous Work.....	10
1.6. Methodology.....	14
1.7. Glossary.....	16
2. Project Planning.....	18
2.1. Milestones, WBS, and Responsibilities.....	18
2.2. Project Scheduling.....	20
2.3. Feasibility Study and Cost Estimation.....	22
2.4. Risk Management.....	25
3. Systems Analysis.....	26
3.1. Business Models.....	26
3.1.1. AS-IS Model.....	26
3.1.2. TO-BE Model.....	27
3.2. Stakeholders Identification.....	29
3.3. VORD Method Requirements Determination.....	31
3.3.1. Brainstorming.....	31
3.3.2. Hierarchy Diagram.....	32
3.3.3. View Point and Service Templates.....	33
3.4. Requirements Definition.....	35
3.4.1. Functional Requirements.....	35
3.4.2. Non-Functional Requirements.....	37
3.5. Process Specifications.....	38
3.5.1. Decision Tree.....	38
3.6. Documenting and Modeling Requirements.....	39
3.6.1. Use-Cases.....	39

3.6.2. Data Dictionary .....	42
4. Architectural Design .....	44
4.1. System Structuring.....	44
4.1.1. Repository Model.....	44
4.1.2. Abstract Machine Model.....	46
4.1.3. System Organization Diagram.....	48
4.1.4. Hardware Organization Model.....	49
4.2. Control Models .....	51
4.2.1. Sequence Diagram.....	52
4.3. Modular Decomposition .....	53
4.3.1. Data Flow Diagrams .....	53
4.3.2. Object Model Diagram.....	60
4.4. Database Design.....	61
4.4.1. Entity-Relationship Model.....	61
4.4.2. Entity-Relationship Diagram .....	62
4.4.3. Database Objects List .....	63
Appendix A: Requirements Gathering Questionnaires.....	69
A.1. Questionnaire 1 .....	69
A.2. Questionnaire 1 Results .....	71
A.3. Questionnaire 2 .....	74
A.4. Questionnaire 2 Results .....	76
Appendix B: Prototype Evaluation.....	78
B.1. Team Evaluation Results.....	78
B.2. Sponsor Evaluation Results.....	88
Appendix C: Software Testing .....	90
C.1. Test Case Results .....	90

Appendix D: Help Documentation .....	99
Appendix E: Source Code.....	117
E.1 Main Screen .....	117
E.2 About Screen.....	120
E.3 Customer Lookup Screen.....	120
E.4 New Customer Screen.....	124
E.5 New Vehicle Screen .....	125
E.6 Pre-Report Screen.....	127
E.7 STS Database Utilities.....	131
E.8 General File .....	132
E.9 Employee Labor Report.....	133
E.10 Financial Report .....	135
E.11 Invoice .....	136
E.12 Services/Sublets Report.....	137

## **1. Introduction**

### **1.1 Abstract**

Many companies' main *product* is the service that they provide their customers. For these companies it is important for them to be able to organize information related to their customers, their vendors and the products that those vendors supply, and most importantly the service that this particular company provides. Most of these service-oriented companies have been around for a while, and maintain most or all of their information on paper, stored in files.

The project that this team will be completing is a software package for service-oriented companies to use as a management tool for all of their data processing and storage. For this particular project our direction will be headed towards automobile service stations. Our sponsor will be *Heights Service Station* in Midland Park, NJ. All information will be stored in a database, which will enable the front-end application to provide simple access to all the necessary information, as well as provide reports that will

act as invoices, customer activity statements, and accounting information. The application that will be produced will be a **Service Tracking System**, and its main purpose will be to give the company a place to store all information on customers, vendors & parts, and the services that the company provides.

The original design of this project will call for the front-end program to be written in Visual Basic, using the supplied ADODB middle-tier utilities for communicating with the database. Given the use of ADO for database connectivity the actual database is somewhat irrelevant for design purposes as long as ADO has a provider for that database. However, as far as implementation goes, an MS Access database will be sufficient for the scope of this project. It should be important to note that were this product to become marketed for actual customers, the database choice is very important, (in these cases the database should probably be something along the lines of MS SQL Server, MySQL, or Oracle, etc.).

## **1.2 Team Members and Job Descriptions**

### **Marc DeFilippo – System Analyst**

- Requirements gathering and documentation. (via sponsor and literature review). Also create relevant structure charts and DFD's of product.
- Analysis at milestones to assure product is meeting requirements.
- Assist other team members with user manuals and help documentation.

### **Partha Majumdar – Database Designer**

- Design & implement database layout based off previous service tracker design and with input from Richard.
- Aid Richard with feasibility study: benefit/cost estimation and risk analysis.
- Design necessary *canned* end-user reports. (Invoices, accounting info, etc.)
- Development of involved test cases that will be used for testing end product.

### **Rima Patel – Front-End Designer**

- Work with Richard in designing the front-end.
- Implement the front-end, most likely using Visual Basic.
- Aid programmers with providing functionality for these VB forms.

- Help with testing product.

**Richard Sonderfan – Project Manager**

- Coordinate all member activities and time usage of team.
- Help to educate the team on the project at hand, aid in the solution of any problems the team is having.
- Act as chief programmer for the implementation of the overall product, and assist in front-end design and implementation.
- Compile all deliverables and progress reports for submission to the professor.

**Vidhu Sukheja – Programmer/Help Author**

- Provide functionality to front-end screens designed by Rima.
- Write the user manual and help files.
- Aid Tony with testing of the finished product.

**Tony Yang – Programmer/QA Engineer**

- Possible product website.
- Provide functionality to front-end screens designed by Rima.
- Aid Partha in design of end-user database reports.
- Full testing of the product, (involving installation, validation of help documentation, and front-end use/functionality).

### **1.3 Background**

There are a number of service-oriented businesses in the world today. While there is much variety among these different businesses, they all provide a service and for the most part manage the same type of information. These companies maintain a base set of information such as customers, vendors and products that those vendors produce, employees, and the service that they provide for their customers. The information that these companies maintain is vital to the success of the company. Their records are what they use to store a history of all customers, services done, and products that they used in doing those services. They will need this information for many different reasons, including keeping track of their financial situation, contacting customers for future service, and as evidence of services done for possible legal reasons, as well as a host of other needs. The customer gets a copy of this information for each service they receive, usually in the form of an invoice or receipt (usually in paper-form).

Many of these companies have been around for a long time, and some are new companies that are trying to get a hold of the market in their service area. Their vital information can be stored in different ways. Large companies in this current day most likely have computerized systems that are used for printing invoices, tracking customers, and updating information with satellite sites around the country or world. However, some smaller service oriented businesses still keep track of all their important information on paper. This is usually because that system works for them due to their relatively small customer and service base, and/or because they cannot afford some of the expensive systems that are out on the market.

While there are many service-oriented companies, focusing on one type of service will ensure that very specific needs are taken care of. This project will deal with one type of service company – an automobile service station. The sponsor for this project is Heights Service Station in Midland Park, NJ. This service station is a small privately owned automobile service shop that is currently managing all of its dynamic data via paper records and filing cabinets.



## 1.4 Problem Statement

It has been proven that the data that a service station generates and maintains is an essential asset to that company. Without it, the company cannot function properly, grow, or properly track their resources. This is where *Service Tracking System* (STS) comes in. This software system will be a service station management solution for relatively small businesses. The reason for targeting smaller businesses is that there already exists solutions for larger companies, and there are many high-end features that larger companies require that would demand more development time than this school project provides.

The STS will handle all major areas of a small automobile server station's information management such as data on employees, customers and their vehicles, vendors and their parts, general station information, service records/invoices, and payment tracking. This system should enable small businesses to maintain all of their information electronically. Having the information stored electronically in the form of a relational database will provide the ability to extract complex reports on any combination of information (customer, billing, company profits, etc.). There is also the added benefit of no longer maintaining large piles of paper records that could get mixed up, lost, or even damaged by some sort of small natural disaster. The sponsor of this project, Heights Service Station, currently manages all of its information via paper records so this software could help out enormously on that point alone.

To make the system easy to use it will be implemented as a Microsoft Windows® application, as this operating system (OS) is most likely one of the easiest to use by non-technical people. Also, Heights Service Station currently has a single-PC setup running

this OS. As stated previously, the actual data storage for this product will be via a database. Using a database means that the data will be more than just stored, it will be stored in relational tables with links to other records and supported by middle-tier interfaces that can easily extract valuable information from the database.

The problem statement is that of management of an automobile service station's information. The solution will be a software system that will provide easy, flexible, and robust access to this information. The software system design will be comprised of a three-tier approach that will be implemented in two actual components: a Microsoft Windows® front-end application, and a relational database for the back-end and information storage. The front-end will be tier three, with tier two (the database interface) built into it. Tier one will be the actual database itself. See figure 1 to illustrate this point.

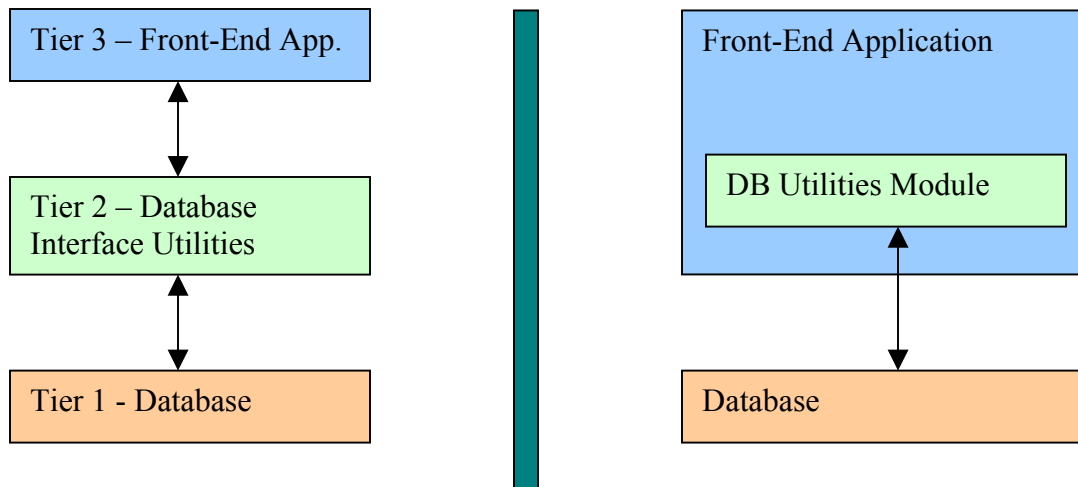


Figure 1.1 This is a diagram of the system design, as opposed to its implementation. The left side of the diagram is the overall design of the system, and the right side is the implementation.

## 1.5 Previous Work

It is important to look at previous work in this area for many reasons. Learning from these software examples will help STS become an *experienced* piece of software that has learned from the mistakes of other companies. Also, we must not implement a software product that does exactly what another product already accomplishes – we must analyze what others have done to make our product both unique and beneficial to the user. By studying these other (similar) software products and receiving direct requirements information from the sponsor, STS can be a very valuable system.

The first product that will be looked at is **TireMax**. This product is actually an Internet-based system that is geared towards tire dealers and their shops. While this is not exactly applicable to the service station model, it is close. This system has some features that will obviously not be used by STS, but the overall structure will be. The TireMax software provides the ability to manage customer and vehicle information, as well as provide a competent point of sales interface (invoicing, etc.). They also provide for the user a set of reports that can be used to analyze information from the database such as sales info and outstanding balances. This seems like a decent software system, but it is ultimately not directly applicable to STS.

([http://www.asatire.com/tiremax\\_pos.htm](http://www.asatire.com/tiremax_pos.htm))

Another product is **Wrenthead Pro**. This product is a newer software solution to service stations. Wrenthead Pro has a very nice graphical interface. We will probably create one similar to this. It uses a lot of icons to make the interface more interesting and easy to use. However, there are some unnecessary functions that will make it too complicated for a user who has no computer related background. This software package

is very comprehensive and is geared towards midsize to large companies. They offer the standard customer/vehicle/invoice tracking as the other software products. One of the largest attractions to this solution is that they have an online parts list and pricing synchronization feature (purchasable as a subscription) that allows the service station to get live updates of parts pricing, etc. In that sense, this product focuses heavily on the parts and inventory end of service station management, whereas STS will focus more on the services provided by the company and the customers receiving those services. It is also important to note here that a large, real-time parts list is beyond the scope of a single semester project. This software also focuses heavily on employees and their times (timecard analysis, etc.).

(<http://www.wrentheadpro.com>)

AutoPro is a software company that makes solutions for automobile stations. One of their products is titled **AutoMotive**. This software has many great functions. However, those functions are not easy to find. The user will need to spend a great deal of time learning and getting used to this software. This product seems to focus greatly on customization and user defined reports, etc. While this is a powerful way to design a software system, it is not necessarily beneficial to a small automobile service station owner who knows little about using computers in the first place.

(<http://www.personal.u-net.com/~autopro/>)

The last shop management software reviewed is InvoMax. InvoMax has a great graphical interface. It gives many details on each function through use of a help system. However, it seems like there is too much information to be entered in the invoice, and we all know that auto shops can be very busy. It is good to have control of inventory, but a

complete parts database may not be necessary. Graphically the invoice here is the direction that STS should be headed. The functionality of this software product is similar to that of what is in store for STS. InvoMax includes an entire parts database, however, which is not necessary for our purposes. An image of the InvoMax screen has been included to illustrate one possible invoice design that may make its way into STS, (image taken from: <http://www.invomax.com>).

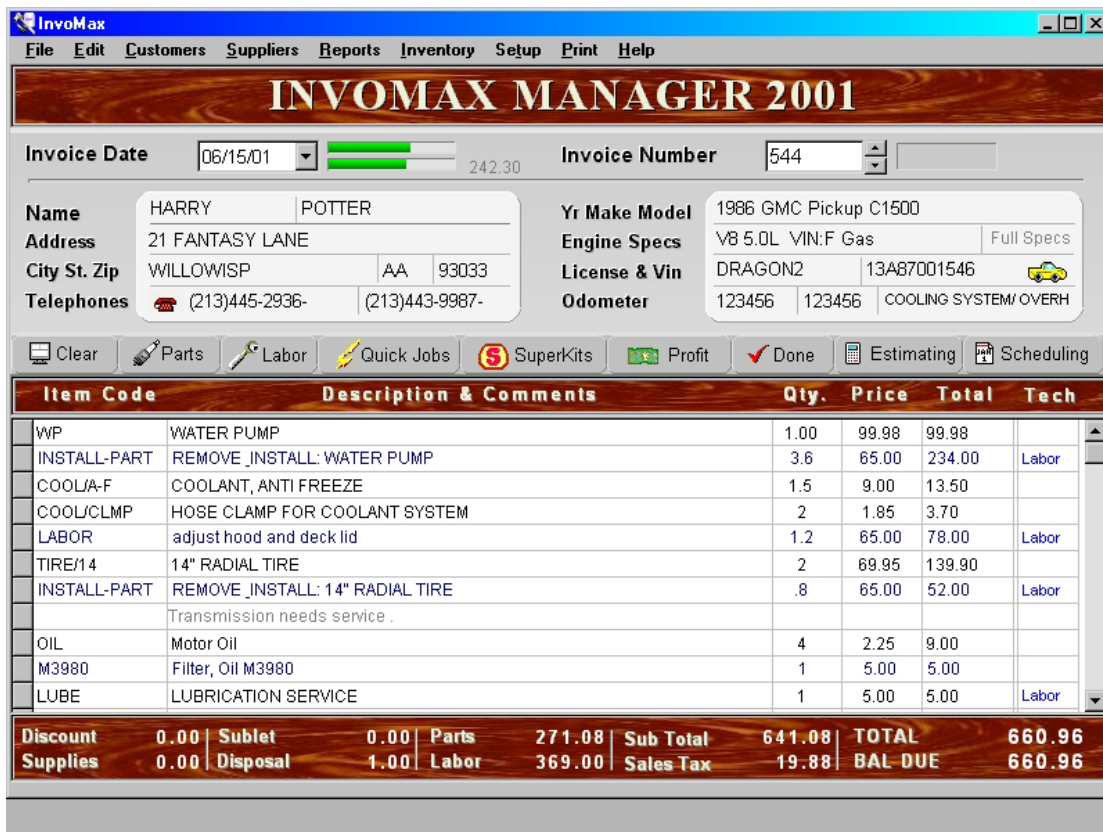


Figure 1.2. The InvoMax invoice screen.

In closing, there are already many products in the market that are similar to STS. The automotive shop management concept is still an idea that is *up for grabs* as there are no *perfect* solutions out there. Also, while some of these reviewed solutions were comprehensive, our goal here is to target smaller businesses that do not require the complexity and depth that some provide. We need to focus on bringing small service

station shops (like Heights Service Station) out of the *paper age* and into the world of computers, showing them how managing information electronically can be beneficial for the reduction of paper storage and for the data reports and analysis.

## 1.6 Methodology

As with any software project, key methodologies will lend their hand to the overall design and integrity of the system. Methodologies are the tools (conceptual and visual) that will be used to guide and explain the entire development of this software system. The right methodologies means that the necessary information that needs to be relayed between the different engineers is done so in an efficient way. Not only is communication a key component to this software project, but also an understanding of the overall process and the ability to follow that process is vital.

An important component that will be used here is **Data Flow Diagrams (DFD)**. DFDs will be the visual representation of how this system distributes the data from inputs (employees/service men) to outputs (customers/accounting info). These diagrams will be used as the result of the requirements elicitation and analysis process. They will communicate to every person on this project exactly what the goal of the system is, and how it should be implemented.

Another important component to this project is the **Entity-Relationship Diagram (ER-D)**. This diagram is used to show the relationships between the different tables of information in the database. Since the database is the main focus for this project, (the data is what drives this system), understanding the relationships between the complex data sets and carefully relaying that information to the system implementers is vital for a successful piece of software to be written. The ER diagrams of this system will truthfully reflect the logical relationship of data that underlies the automobile service station.

The overall methodology for this project is the software process model that is used to describe and dictate how the software should be researched, designed,

implemented, distributed, and maintained. This model will describe the *life cycle* of this software. The process model for this project will be a **hybrid** of the **Rapid Application Development (RAD)** model and the **Spiral** model. There will be an emphasis on prototyping and the throwaway mentality. What this means is that RAD will be accomplished not by several sub-teams working in parallel, but each individual in the team working in parallel. Unfortunately due to time constraints this means that some processes must start before the requirements documentation is complete (for example). The spiral model is seen here, as we will keep making passes through requirements gathering, redesigning, implementation and prototyping, and getting user feedback on the prototype. This will go on and on, as we need to constantly re-evaluate the software for compliance with the requirements. In the evaluation matrix in table 1, the rationalization of which software process model(s) to choose becomes clear. For our purposes, a hybrid of the RAD and Spiral models would suit this project best.

<b>Process Model</b>	<b>Short-Term Project</b>	<b>Cost</b>	<b>Learning Curve</b>	<b>Technical Risk</b>	<b>Multi-Tier Approach</b>	<b>Total Score</b>
RAD	80	50	35	45	95	350
Spiral	40	80	90	90	65	365
Waterfall	80	70	75	70	40	335

Table 1. 1 Evaluation Matrix for possible software process models. (Scores: 0 to 100)



## 1.7 Glossary

- BEP- Break-even point. This is the point in time of a products life cycle that decreasing costs and the increasing profits intersect.
- DFD - Data Flow Diagrams. These are diagrams used to understand the movement of data (information) within the overall software system.
- ERD – Entity Relationship Diagram. These diagrams are used to show the relationships between the different tables of information in a database.
- GUI – Graphical User Interface. This is what the end-user sees and interacts with to view and enter data into the program. The GUI is the actual visible end of the software.
- Invoice - This is a description of the services and parts provided to a customer. It acts as a receipt and as a record for accounting purposes.
- NPV - Net Present Value (NPV) is a way of comparing the value of money now with the value of money in the future. A dollar today is worth more than a dollar in the future, because inflation erodes the buying power of the future money, while money available today can be invested and grow. (<http://www.finaid.org/loans/npv.phtml>)
- OS - Operating System. This is the main software that controls a PC. An example would be Microsoft Windows®.
- PC- Personal Computer. This is the computer that is being used by the user to run personal software. STS will eventually run on the PC.
- Point of Sale – A one-stop point (or program or system) that employees use to manage their sales.

- Prototyping – The process of creating not fully functional versions of a software system so that end users may test the current version and supply feedback—thus enriching the quality of the software.
- RAD - Rapid Application Development. A method of developing software that causes many processes of development to be happening in parallel, thus speeding up the development process.
- ROI – Return On Investment. This is the amount of money that should be expected back from sales of a product once all costs are paid.
- Service Station – In this case, we are talking about an Automobile Service Station.
- STS - Service-Tracking System. This is the name of the software product to be developed by this team.
- UI – User Interface. This is any type of computer interface that a human uses to either input or receive output from a computer.
- VB – Visual Basic. This is the software programming language that will be used to develop this project.
- WBS - Work Breakdown Structure. This is the breakdown of all of the work that that must be done for the entire project, with dates, and the names of those responsible for each one.

## 2. Project Planning

### 2.1 Milestones, WBS, and Responsibilities

The milestones table follows. These are the deliverables that will dictate the course of the projects development.

Start Date	End Date	Name	Breakdown	Members Responsible
9/17/2002	9/24/2002	Project Abstract and Team Members.	<ol style="list-style-type: none"> <li>1. Project Abstract</li> <li>2. Members</li> </ol>	Richard
9/24/2002	10/8/2002	Project Planning	<ol style="list-style-type: none"> <li>1. Background</li> <li>2. Problem Statement</li> <li>3. Previous Works</li> <li>4. Methodology</li> <li>5. Glossary</li> <li>6. Milestones, WBS, Responsibilities</li> <li>7. Project Sched.</li> <li>8. Feasibility</li> <li>9. Cost Estimation</li> <li>10. Risk Manag.</li> </ol>	Richard, Partha
9/24/2002	10/8/2002	Prototype #1	Prototype of the user interface.	Richard, Rima
9/17/2002	10/22/2002	Requirements	<ol style="list-style-type: none"> <li>1. Business Models</li> <li>2. Stakeholders</li> <li>3. Gathering</li> <li>4. Documenting and Modeling</li> </ol>	Marc, (Partha)
10/8/2002	10/22/2002	Prototype #2	Prototype of the user interface.	Richard, Rima, Vidhu, Tony
10/22/2002	11/5/2002	Architecture	<ol style="list-style-type: none"> <li>1. System Structure</li> <li>2. Control Model</li> <li>3. Modular Decomposition</li> <li>4. Database Des.</li> <li>5. UI Design</li> </ol>	Parths, Marc, Rima.
10/22/2002	11/5/2002	Prototype #3	Prototype of the user interface.	Richard, Rima, Vidhu, Tony
11/5/2002	12/10/2002	Final Presentation	<ol style="list-style-type: none"> <li>1. All documents.</li> <li>2. Working software with full help.</li> </ol>	All Members

Table 2.1 Milestones.

The work-breakdown structure for this project is shown in the following table. The ID for each task is used to map to the different items in the Gantt/Pert charts.

ID	Task Name	Duration	Start Date	Finish Date	Resource Names
1	D1 - Project Abstract & Team	1 day	9/24	9/24	Richard
2	D2 - Progress Report to professor	1 day	10/1	10/1	Richard
3	Requirements Questions	8 days	9/24	10/1	Marc
4	Feasibility, Benefit/Cost estimation and risk outline	11 days	9/24	10/4	Partha
5	Project Planning	14 days	9/24	10/7	Richard
6	PowerPoint Presentation Created	0.5 days	10/8	10/8	Vidhu, Richard
7	D3 - Project plan and feasibility report & presentation	0.5 days	10/8	10/8	Richard, Tony
8	Literature review completed	5 days	10/4	10/8	Marc, Vidhu, Tony
9	Mockup or screenshots of front-end	15 days	9/24	10/8	Rima
10	Final database layout	18 days	9/24	10/11	Partha
11	D4 - Progress Report to Professor	1 day	10/15	10/15	Richard
12	Requirements Report 1st draft	6 days	10/10	10/15	Marc
13	Initial VB front-end forms	7 days	10/9	10/15	Rima
14	PowerPoint Presentation Created	6 days	10/16	10/21	Vidhu, Marc
15	D5 - Final draft of requirements report	7 days	10/16	10/22	Marc, Rima
16	Completed source code example	7 days	10/16	10/22	Richard
17	D6 - Progress Report to Professor	1 day	10/29	10/29	Richard
18	Architectural design 1st draft	18 days	10/12	10/29	Marc, Partha
19	D7 - Final draft of Architectural Report and Presentation	7 days	10/30	11/5	Partha, Vidhu
20	D8 - Progress Report to Professor	1 day	11/12	11/12	Richard
21	D9 - Progress Report to Professor	1 day	11/19	11/19	Richard
22	SQL queries	6 days	11/19	11/24	Partha
23	Prototype given to sponsor	1 day	11/23	11/23	Marc, Richard
24	Software Complete (initial version)	33 days	10/23	11/24	Vidhu, Tony, Rima, Richard
25	Outline for user manual and help	8 days	11/17	11/24	Marc
26	Testing complete	7 days	11/27	12/3	Tony, Rima, Richard
27	Changes made to software based on sponsor input	10 days	11/24	12/3	Richard, Tony
28	User manual and help files complete	9 days	11/25	12/3	Vidhu, Partha
29	D10 - Progress Report to Professor	1 day	12/4	12/4	Richard
30	Sponsor Mail Evaluation	2 days	12/5	12/6	Richard
31	PowerPoint Presentation Created / Final Documentation complete	10 days	11/24	12/3	Marc
32	Meet in class to review all materials	1 day	12/3	12/3	(Everyone)
33	Meet at Library for Presentation practice	1 day	12/8	12/8	(Everyone)
34	D11 - Final presentation & product delivery	1 day	12/10	12/10	(Everyone)

Table 2.2 Work-Breakdown Structure (WBS)

## 2.2 Project Scheduling

This section contains the Gantt charts and Pert (Network) charts of the project timeline. These charts will help to illustrate how the project load will be distributed over time. Note: the weeks in the Gantt charts are divided up in groups of seven days, starting with Tuesday.

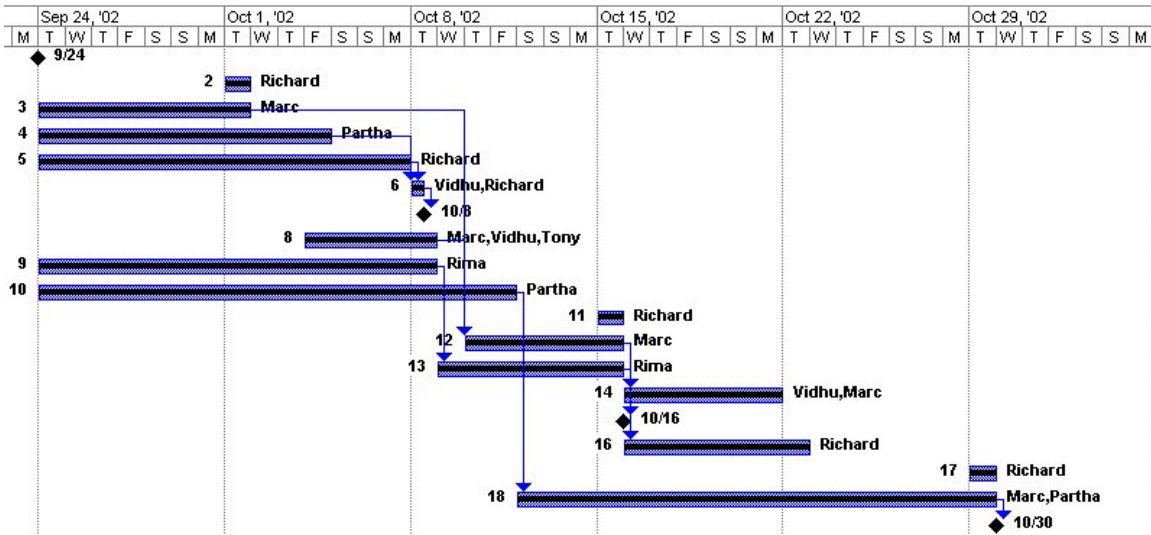


Figure 2.1 Gantt Chart: First half of the project. (September 24 – November 4)

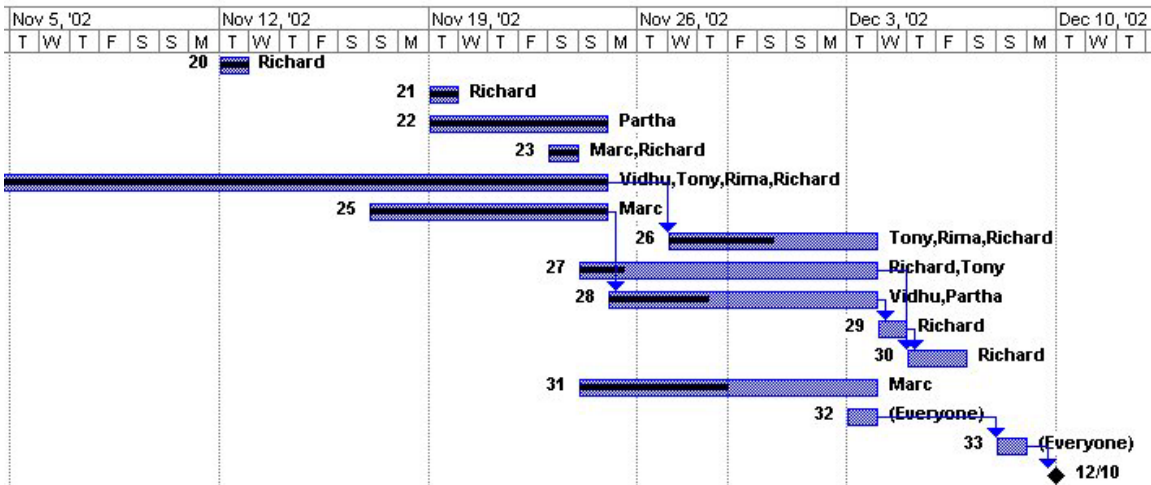


Figure 2.2 Gantt Chart: Second half of the project. (November 5 – December 10)

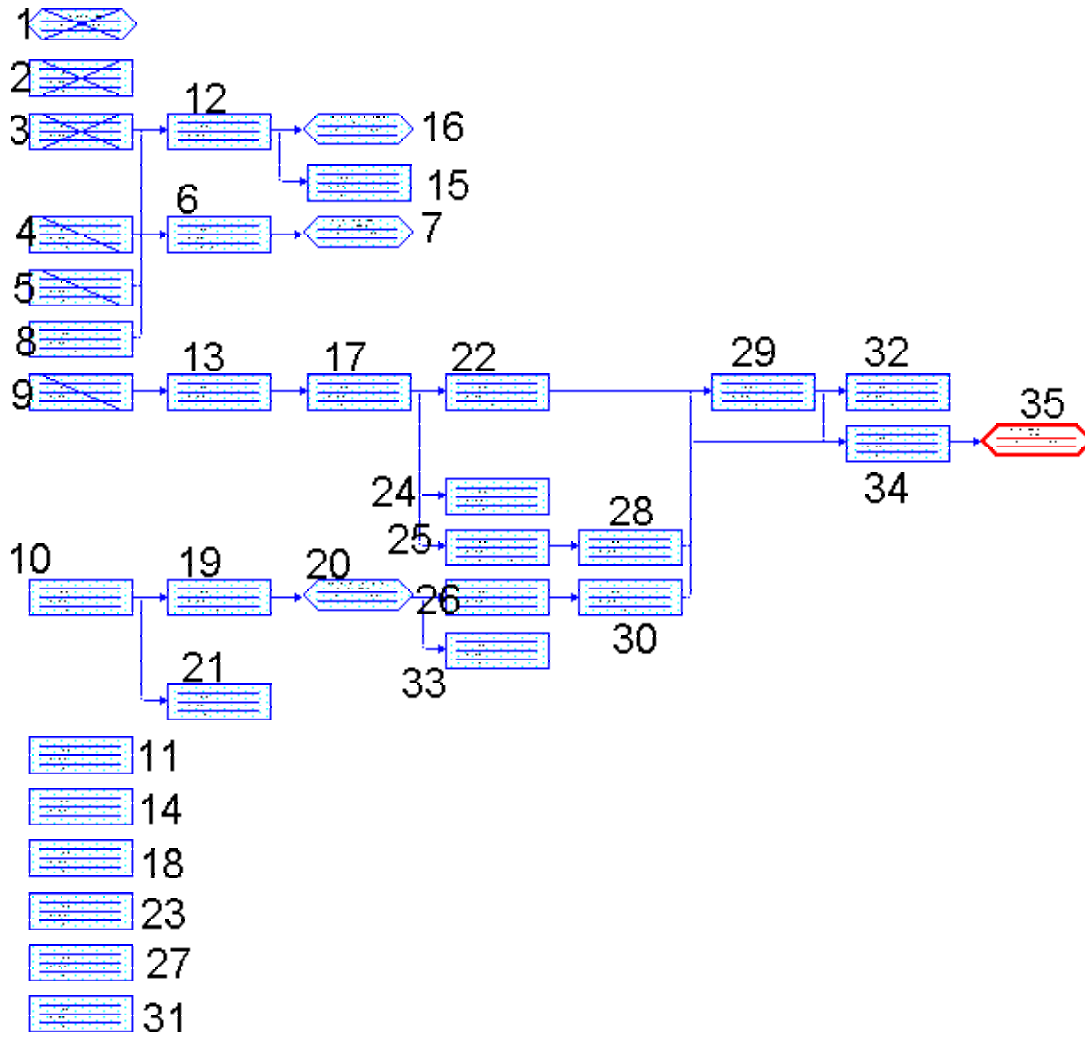


Figure 2.3 Pert (Network) Chart: Shows critical paths and bottlenecks in the WBS.

## 2.3 Feasibility Study and Cost Estimation

<b>Tangible Benefits Worksheet</b>	
Service Tracking System	
Year 1 through 5	
A. Cost reduction or avoidance	\$10,000
B. Error reduction	\$5,000
C. Increased flexibility	\$12,000
D. Increased speed of activity	\$20,000
E. Improvement in management planning and control	\$7,000
_____	_____
	\$54,000

Table 2.3 Tangible Benefits Worksheet.

<b>One-Time Costs Worksheet</b>	
Service Tracking System	
Year 0	
A. Development Costs	\$30,000
B. New Hardware	\$15,000
C. User Training	\$7,000
D. Installation and Setup (Data and Software)	\$10,000
E. Other	\$0
_____	_____
	\$47,000

Table 2.4 One-Time Costs Worksheet.

<b>Recurring Costs Worksheet</b>	
Service Tracking System	
Year 1 through 5	
A. Software application maintenance	\$5,000
B. Annual database support	\$5,000
C. Annual hardware support	\$3,000
D. Incremental data storage	\$2,000
E. New software	\$1,000
F. New Hardware	\$2,000
G. Other	\$0
<hr/>	
	<b>\$18,000</b>

Table 2.5 Recurring Costs Worksheet.

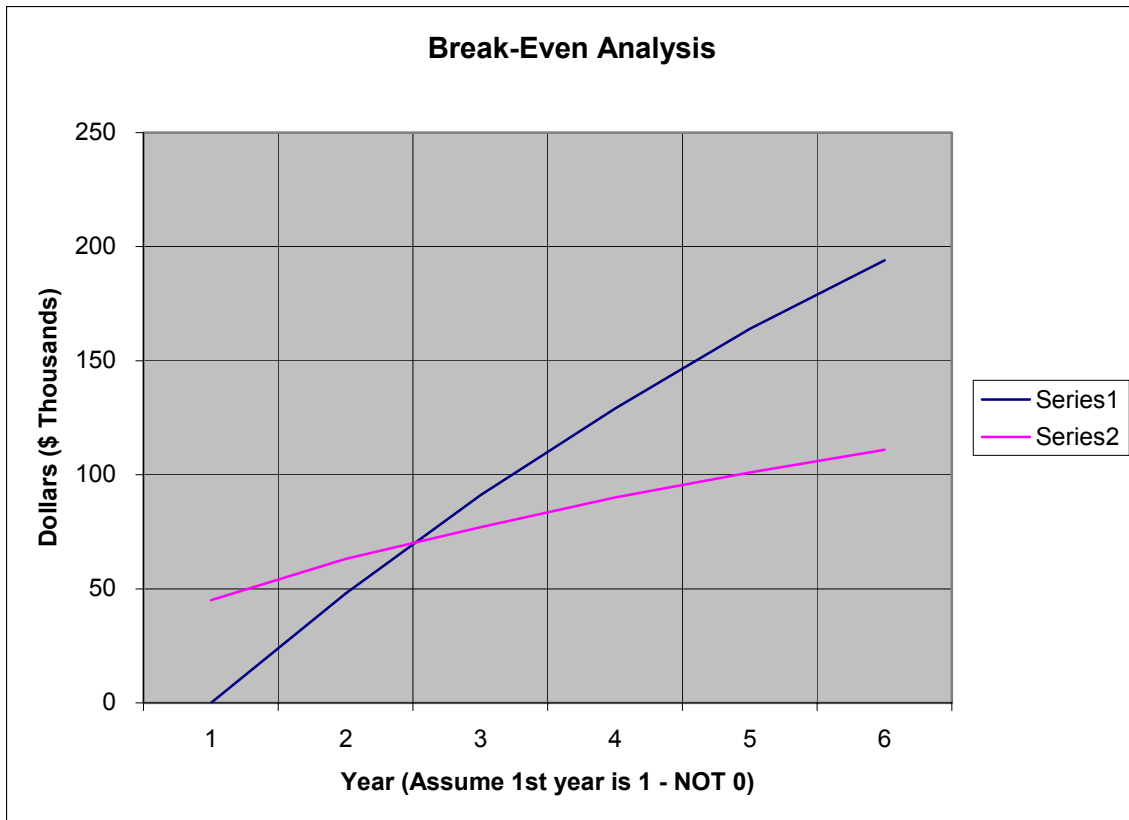


Figure 2.3 Break Even Analysis. The BOP is about 2 years.



<b>Economic Feasibility Analysis</b>							
<b>Service Tracking System</b>							
	Year 0	Year 1	Year 2	Year 3	Year 4	Year 5	Totals
Net economic benefit	\$0	\$54,000	\$54,000	\$54,000	\$54,000	\$54,000	
Discount Rate (12%)	1	0.8928	0.7971	0.7117	0.6355	0.5674	
PV of benefits	\$0	\$48,214	\$43,048	\$38,436	\$34,318	\$30,641	
<b>NPV of all benefits</b>	\$0	\$48,214	\$91,263	\$129,699	\$164,017	\$194,658	\$194,658
<b>One Time Costs</b>	(\$47,000)						
Recurring Costs	\$0	(\$18,000)	(\$18,000)	(\$18,000)	(\$18,000)	(\$18,000)	
Discount Rate (12%)	1	0.8928	0.7971	0.7117	0.6355	0.56746	
PV of Recurring Costs	\$0	(\$16,071)	(\$14,349)	(\$12,812)	(\$11,439)	(\$10,214)	
<b>NPV of all Costs</b>	(\$47,000)	(\$63,071)	(\$77,421)	(\$90,233)	(\$101,672)	(\$111,886)	(\$111,886)
<b>Overall NPV</b>							<u>\$82,772</u>
<b>Overall ROI - (Overall NPV / NPV of all Costs)</b>							<u>0.74</u>
<b>Break-even Analysis</b>							
Yearly NPV Cash Flow	(\$47,000)	\$32,143	\$28,699	\$25,624	\$22,879	\$20,427	
Overall NPV Cash Flow	(\$47,000)	(\$14,857)	\$13,842	\$39,466	\$62,345	\$82,772	
Project break even occurs between years 1 and 2							
<b>Actual break even occurred at year:</b>	<b>1.52</b>						
NOTE: All dollar values are rounded to the nearest dollar							

Table 2.6 Economic Feasibility Analysis.

## 2.4 Risk Management

### Project Size Risk: Medium

- There are 6 members in the team – adequate for the project to complete on time.
- Duration of the project is 3 months – leaving the team extremely marginal time to analyze, design, develop, test, implement and document the system.
- Number of organizational departments is one, however, given the user infrastructure it is neither necessary nor desirable.
- Size of the programming effort in terms of hours is approximately 100 for each of the members of the team totaling 600 hours. This is adequate for the size of the project even though none of the developers in the team are solely dedicated to this project on a daily basis.

### Project Structure Risk: Medium

- This will be a new system making it a new territory for both the user and the development team.
- There will be some business procedural changes resulting in some challenges for the user.
- User perceptions are clear resulting in enthusiastic participation.
- Management commitment to the system has resulted in greater cooperation and volunteered information. This makes it extremely beneficial to the requirements gathering process.

### Development Group Risk: Low

- Majority of the members of the development group is familiar with the tools chosen for which the individual is responsible to build the system.
- The team is becoming familiar with the application through requirements analysis and through discussions.
- Some research into competitor's products has made the team familiar with similar systems although of different size.

### User Group Risk: Low

- The user is not currently familiar with the information systems development process.
- The user is familiar with proposed application area making it easier to gather requirements.
- The user is not familiar with using similar system, however, adequate training and documentation will be provided.

### **3. Systems Analysis**

#### **3.1 Business Models**

##### **3.1.1 AS-IS Model**

Heights Service Station's current management system is a paper one. Invoices are written on a triplicate form. The customer takes one copy, another copy is stored by month for sales tax records, and the third copy is filed alphabetically by year. Filing cabinets on the premises can store six years worth of invoices.

This current system is extremely inefficient when it comes to the time needed to access and compile data. Reconstructing the work history for a particular customer requires searching through files to find all of their invoices. The fact that a single customer's records may be spread over several years only adds to the work needed to find them. This situation occurs often; for work history, warranty purposes, or invoice duplication. Past invoices are also used to compile the sales tax figures for gross income reporting, which is utilized by the company's accountant. There is also a great amount of redundant data, since each invoice produces two copies to be stored.

It is clear that the speed of data access, filtering, and combining is a major problem with the current management system at Heights Service Station. The following diagram illustrates this system.

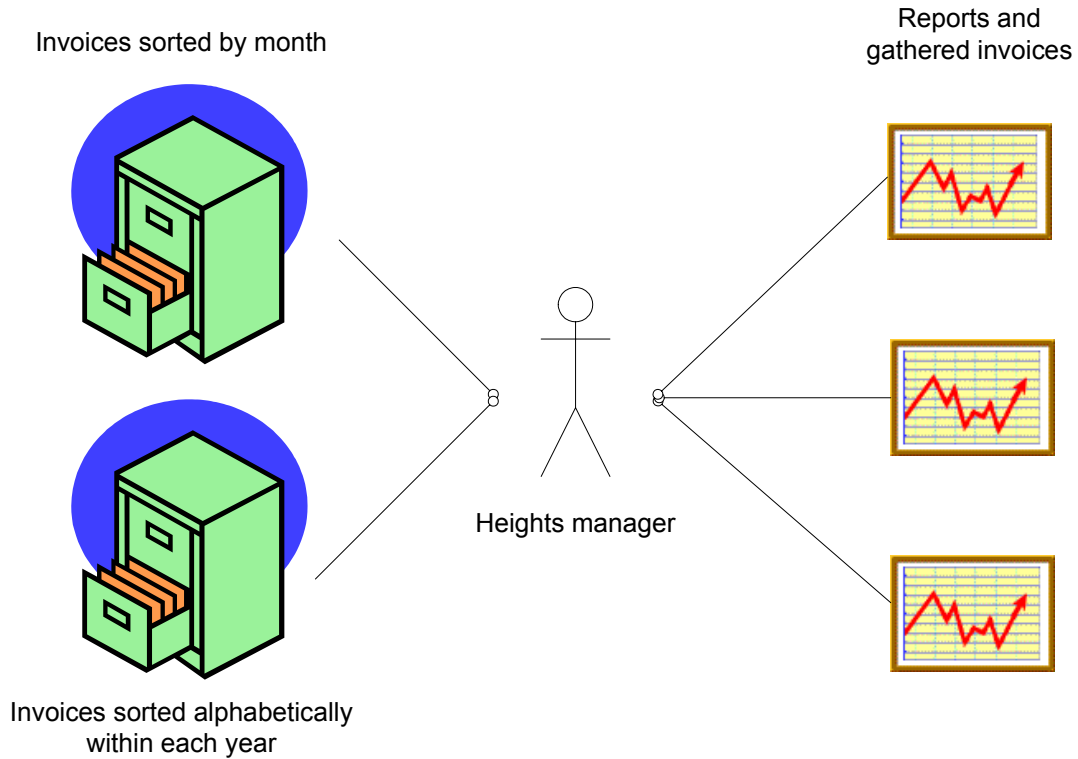


Figure 3.1 AS-IS Model

### 3.1.2 TO-BE Model

Service Tracking System will solve the problems of the current paper system by automating tasks, storing information more logically, performing searches on various criteria, and combining data. Technically, this system will perform all the functionality that a DBMS offers, such as report generating, data entry via forms, and complex querying, as well as a custom tailored interface to meet this client's specific needs.

The new system is entirely computer based and will run on a single PC at Heights Service Station. Creating an invoice will only involve entering the relevant information into the invoice form of the program. A returning customer's information (address, phone

numbers, etc.) will already be filled in by the system, as well as any other commonly used information. The invoice will be printed out for the customer while a record is stored in the system. All invoices can be searched by invoice number, customer name, or date, (to name a few examples), eliminating the time and limitations of the filing cabinet system. A complete work history can be found quickly and easily simply by searching the system for all invoices of a particular customer. End of year reports can be generated automatically based on stored data. Work suggestions can be created “on the fly” when certain criteria are detected. Also redundancy and extra work is eliminated because data is only stored once and retrieved when needed. The following diagram illustrates this improved system.

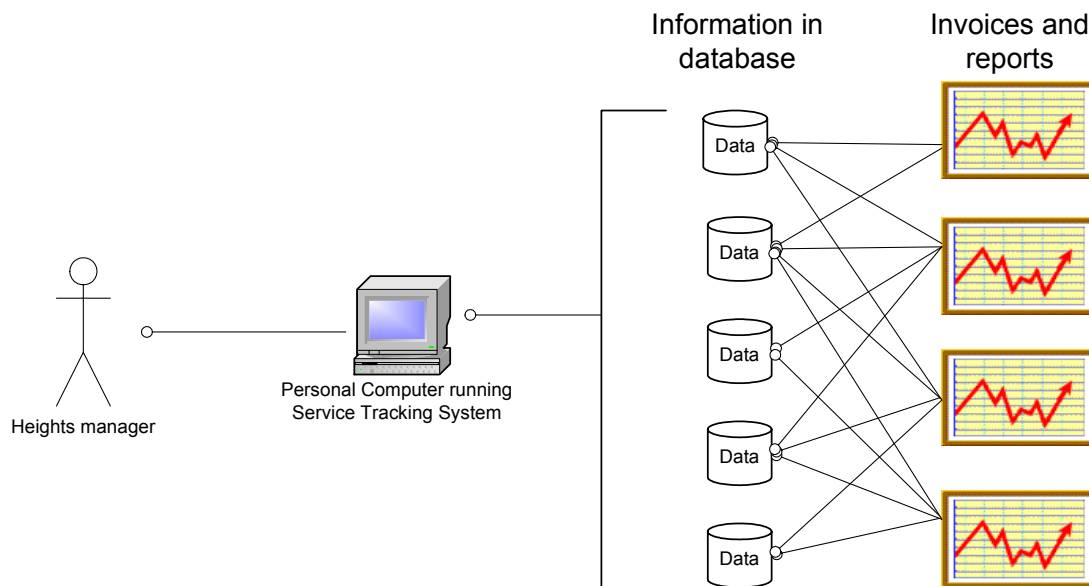


Figure 3.2 TO-BE Model

## 3.2 Stakeholders Identification

Service Tracking System has four stakeholders:

- Owner
- Customers
- Accountant
- Employees

**Heights Service Station's owner** will be the primary user of STS. To him, the interface should be clear and comprehensible. The more intuitive and natural the functions become, the quicker and easier it will be to create an invoice, search for records, or determine a warranty period. When performing these tasks for a customer, a speedy retrieval will be greatly appreciated.

STS should provide all the functionality needed to accomplish the day-to-day business tasks as well as the end of term report generating. Since information can no longer be physically searched for in filing cabinets, the system must be able to perform any operation needed, or its advantage to Heights over the paper system will disappear.

To the owner of Heights, data integrity and accuracy are of great importance. Since STS handles invoice and billing information, an error in the data may mean a loss in profits or incorrect reporting.

**Customers** of Heights Service Station can also be directly affected by data integrity. Erroneous data can cause over billing (error in price or calculations), missed calls (error in customer's personal data), installation of the wrong part (error in vehicle

information), or any other number of mistakes and inconveniences. Also, an incorrect work history can greatly affect the outcome of any diagnostic work

The speed and ease of use of STS also affects customers. If tasks can be carried out easily with the system, Heights Service Station will have more time to spend with a customer or work on their vehicle.

**Heights Service Station's accountant** also has a stake in data integrity and report generating. Incorrect sales or tax figures will skew any year-end reporting and filing. The lack of certain report or filtering capabilities by STS will limit or even prevent any accounting tasks to be carried out due to a lack of available data.

**Employees of Heights Service Station** have a concern in whether their information is stored correctly. An incorrect hourly rate or hours worked may cause them to be shorted on their paycheck. Also, missing or incorrect notes about an employee's performance may cause them to miss out on possible benefits or rewards.

### 3.3 VORD Method Requirements Determination

#### 3.3.1 Brainstorming

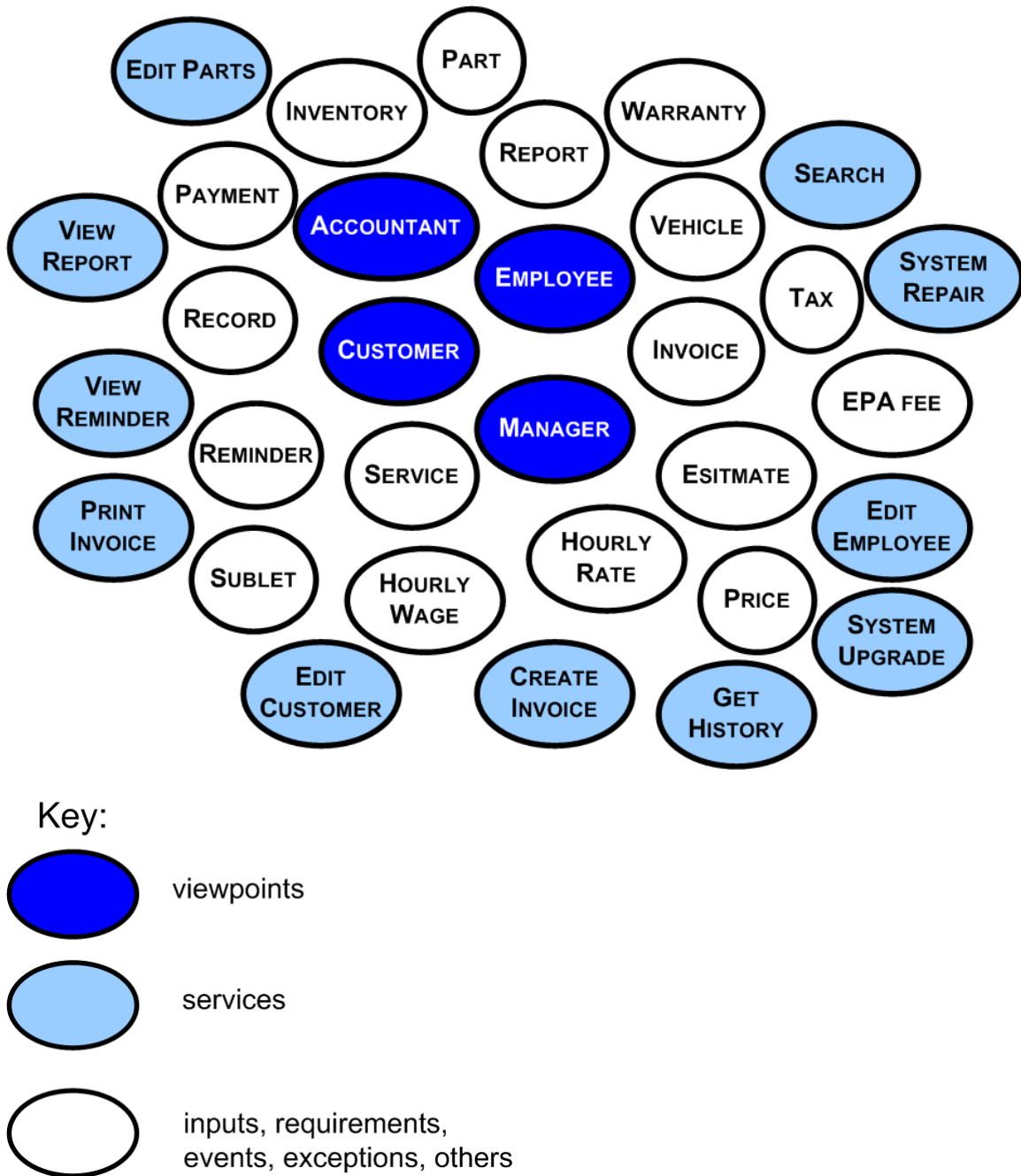


Figure 3.3 Brainstorming



### 3.3.2 Hierarchy Diagram

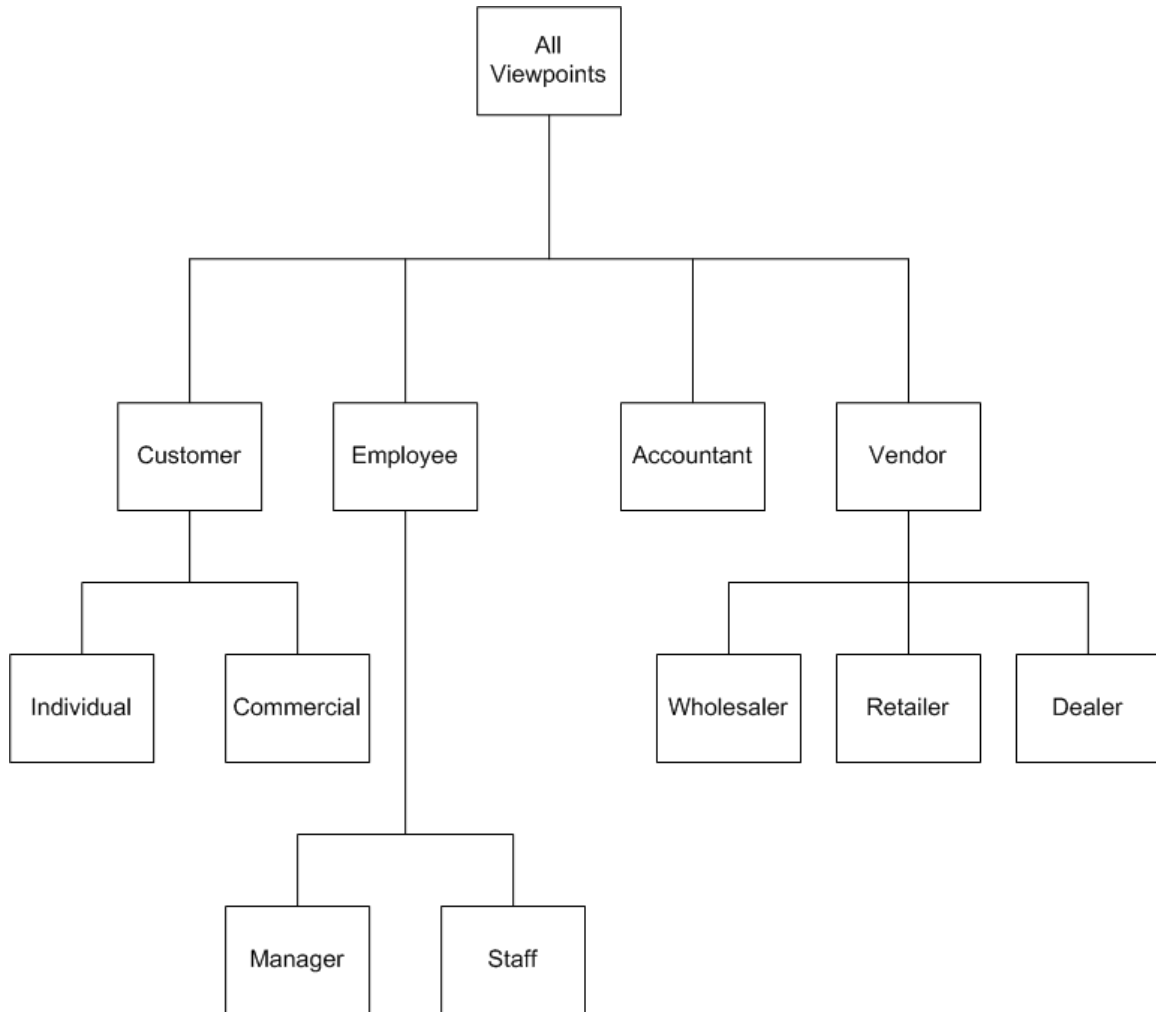


Figure 3.4 Hierarchy Diagram

### 3.3.3 View Point & Service Templates

*Employee Viewpoint:*

<b>Reference:</b>	Employee
<b>Attributes:</b>	Name, Address, Phone#, Wage
<b>Events:</b>	Create Invoice Edit Records Search Records Select Report
<b>Services:</b>	Create & Print Invoice Get History View Reminder Edit Parts, Customer, or Employee View Report Search
<b>Sub-VPs:</b>	Individual Commercial

*Employee Service Templates:*

<b>Reference:</b>	Create & Print Invoice
<b>Rationale:</b>	The most used function of STS. To bill customers.
<b>Specifications:</b>	User selects the Invoice button and enters all pertinent information.
<b>VPs:</b>	Employee

<b>Reference:</b>	Get History
<b>Rationale:</b>	To compile a list of all invoices for a given customer.
<b>Specification:</b>	User selects the History button or Customer button.
<b>VPs:</b>	Employee

<b>Reference:</b>	View Reminder
<b>Rationale:</b>	A system-generated note based on time or vehicle mileage.
<b>Specifications:</b>	User creates an invoice & system generates a reminder.
<b>VPs:</b>	Employee

<b>Reference:</b>	Edit
<b>Rationale:</b>	Allows user to add or modify information.
<b>Specifications:</b>	User selects the appropriate New or Modify
<b>VPs:</b>	Employee

<b>Reference:</b>	View Report
<b>Rationale:</b>	A report based on the selected criteria is used for progress tracking and accounting purposes.
<b>Specifications:</b>	User selects the desired report.
<b>VPs:</b>	Employee Accountant

<b>Reference:</b>	Search
<b>Rationale:</b>	Allows the user to search for a particular invoice, customer, part, vehicle, etc.
<b>Specifications:</b>	User selects Find from the appropriate window.
<b>VPs:</b>	Employee

*Accountant View Point:*

<b>Reference:</b>	Accountant
<b>Attributes:</b>	<i>none</i>
<b>Events:</b>	Select Report
<b>Services:</b>	View Report
<b>Sub-VPs:</b>	<i>none</i>

*Accountant Service Template:*

<b>Reference:</b>	View Report
<b>Rationale:</b>	Allows the user to search for a particular invoice, customer, part, vehicle, etc.
<b>Specifications:</b>	User selects Find from the appropriate window.
<b>VPs:</b>	Employee

## 3.4 Requirements Definition

### 3.4.1 Functional Requirements

1. Service Tracking System must provide areas for the following information on each Invoice:
  - Customer's first name
  - Customer's last name
  - Customer's address
  - Customer's telephone number
    - Work, home, cell numbers
  - Date
  - Invoice number
  - Date of invoice
  - Vehicle make
  - Vehicle model
  - Vehicle year
  - Engine displacement in liters or cubic inches
  - License plate
  - Mileage (in and out)
  - VIN number
  - Service(s) description
  - Hours of labor (per service)
  - Hourly rate (= \$68 currently, but may be changed)
  - Labor total (\$ = hours of labor x hourly rate)
  - Parts list (organized by type)
    - Description
    - Quantity
    - Retail price
    - Warranty information (90 days, 1 year, lifetime, other)
  - Parts total (\$ = parts quantity \* price)
  - Sublet items (work sent out)
  - EPA recycling fees
    - For tires, oil, filters, and antifreeze
  - Subtotal (\$ = labor total + parts total + sublet fees + epa fees)
  - Tax (\$ = [.06 x subtotal])
  - Total (\$ = subtotal + tax)
  - Payment type (cash, check, MasterCard, VISA only)
  - Notes, reminders
  - Heights Service Station warranty ("3,000 miles or 90 days" + parts warranty)
  - Servicing technician
  - Parts statement ("All parts new unless otherwise specified")
  - Authorization agreement with room for customer signature

2. Service Tracking System must allow storage for the following data for each Part:
  - Part number
  - Brand
  - Supplier's invoice number
  - Is the part new or reconditioned
  - Warranty information
    - Client suggests an asterisk by individual parts on the invoice to reference a warranty statement elsewhere on the invoice
  
3. Service Tracking System must allow storage for the following data for each Customer:
  - Name
  - Address
  - Home, work, and/or cell phone numbers
  - A history of services performed (invoice numbers)
  - Recommendations for future services
  
4. Service Tracking System must allow storage for the following data for each Employee:
  - Name
  - Address
  - Phone numbers
  - Achievements
  - "screw-up's or come-back's"
  - Tax withholdings
  - Vacation days allowed
  - Vacation day taken
  - Deductions
  - Pay roll info
  
5. Service Tracking System must allow storage for the following data for each Supplier:
  - Name
  - Phone numbers
  - Parts and/or services available
  
6. Service Tracking System must generate the following reports:
  - Parts sales per month and year
  - Labor sales tax per month and year
  - Parts sales tax per month and year
  - Total sales tax per month and year
  - Labor hours billed per employee per month and year
  - Sublets information (type and total price)
  - Number of each type of service performed per month and year
  - All invoices for a given month

7. Service Tracking System must provide the function to create an Estimate. This Estimate may be referenced later in the creation of an Invoice.

### 3.4.2 Non-Functional Requirements

DESCRIPTION	REQUIREMENT TYPE
Full documentation and full software product must be completed by December 10, 2002	Delivery requirement
STS must run acceptably on a PC with 128 MB of RAM, and an 866 MHz processor.	Efficiency requirement
STS must run in a Microsoft Windows ME environment.	Portability requirement
STS must store at least the 700 current customer records plus new records.	Space requirement
STS will have no “down time” during the workday for self-maintenance.	Reliability requirement
STS must be designed so that future access to an external parts supplier’s database would be possible.	Interoperability requirement
The GUI of STS must be one such that a novice/intermediate computer user will feel comfortable and effective using the system.	Usability requirement

### 3.5 Process Specifications

#### 3.5.1 Decision Tree

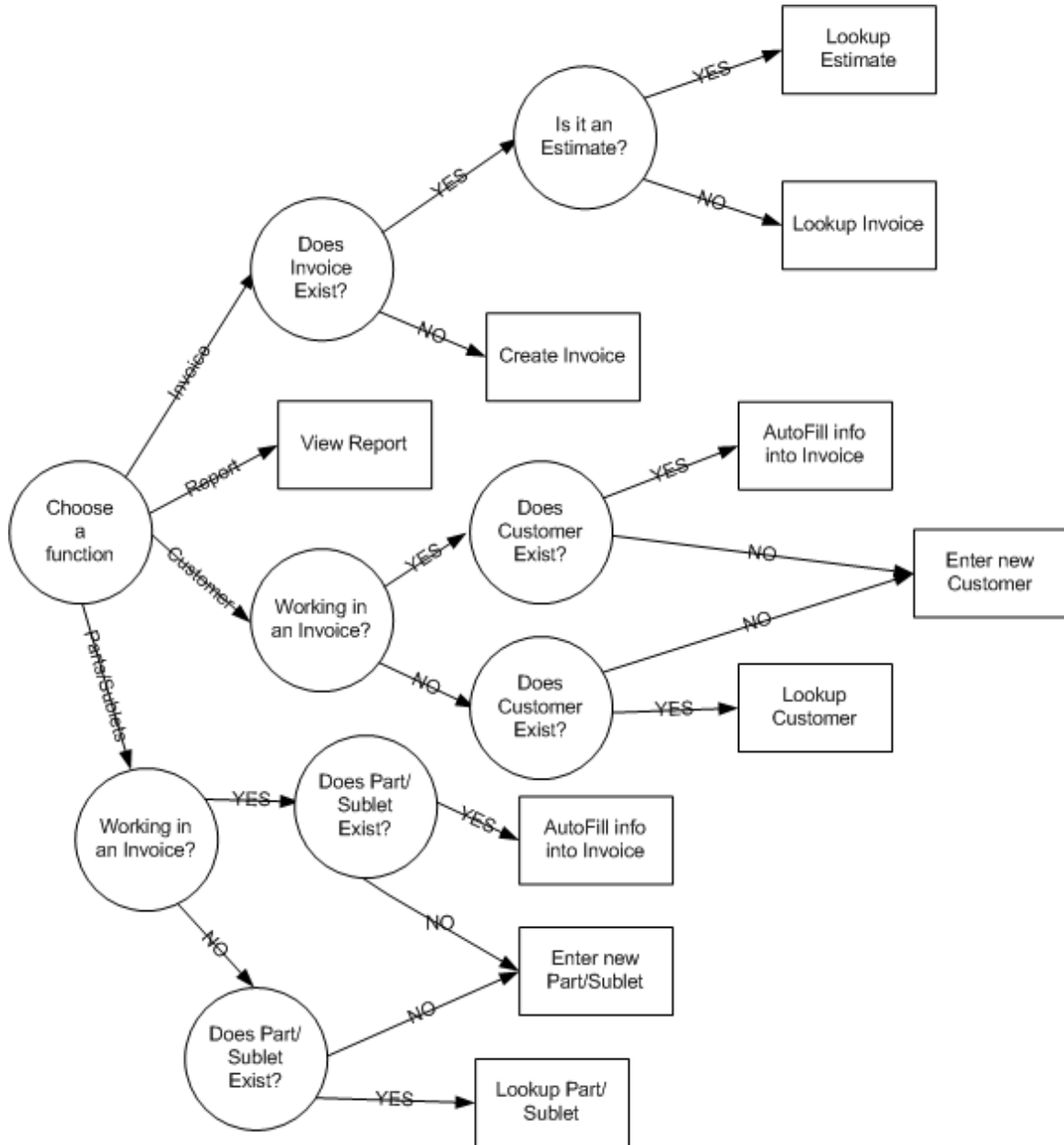


Figure 3.5 Decision Tree

## 3.6 Documenting and Modeling Requirements

### 3.6.1 Use-Cases

Use-Case diagram for Service Tracking System:

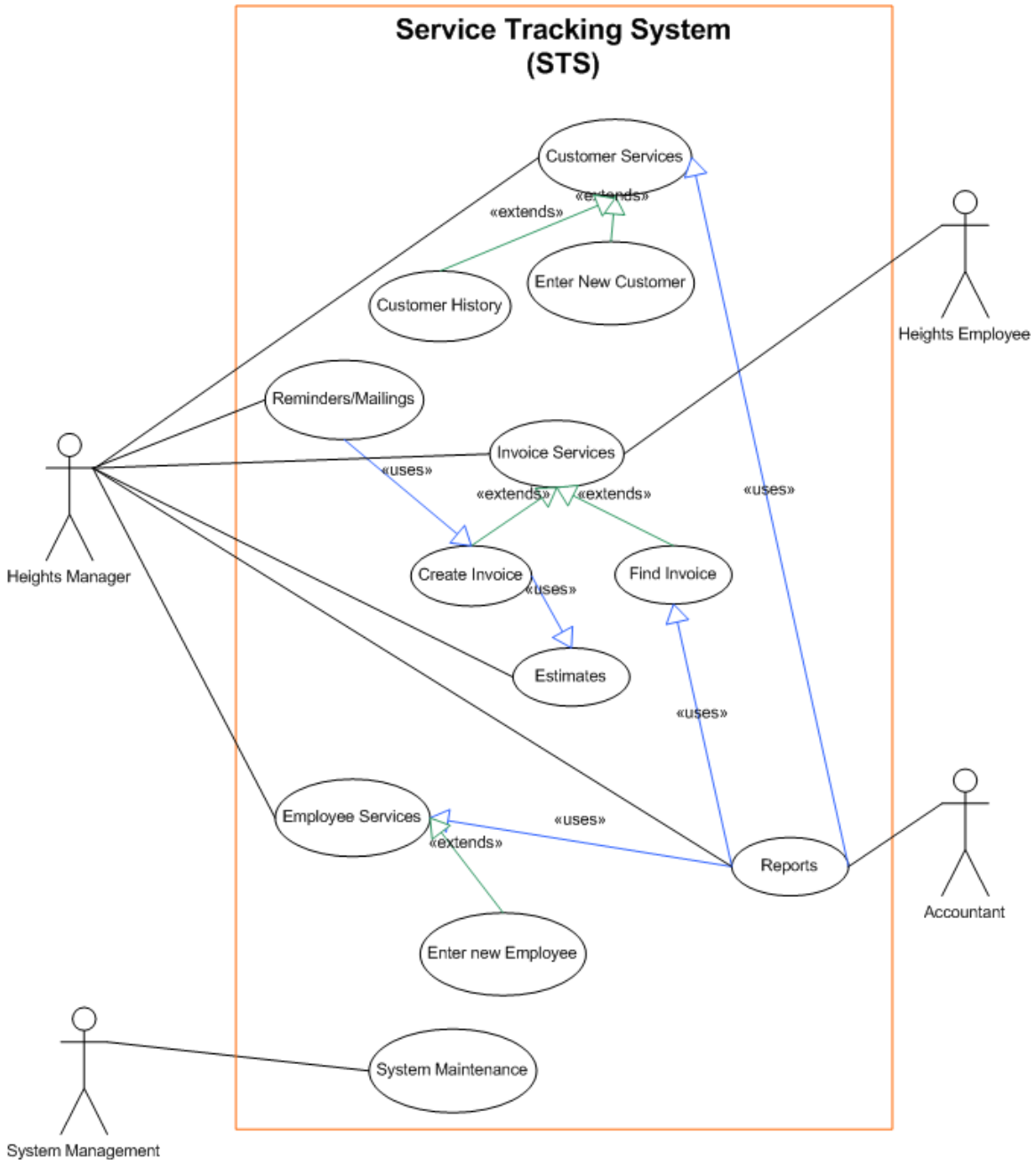


Figure 3.6 Use-Case Scenario



Use-Case descriptions:

<b>USE-CASE</b>	<b>ACTORS</b>	<b>DATA</b>	<b>STIMULUS</b>	<b>RESPONSE</b>
Customer Services	Heights Manager, STS	A collection of functions for dealing with customer data. Entering, editing, and viewing data.	The user selects the desired function from the system interface.	The specific interface for the selected function.
Reminders/Mailings	Heights Manager, STS	STS will generate a work-needed reminder to be placed on an invoice or saved for future mailing.	STS detects a pre-programmed reminder-needed condition.	A reminder is generated and possibly stored in the database.
Invoice Services	Heights Manager, STS, Heights Employee	A collection of functions for dealing with invoices.	The user selects the desired function from the system interface.	The specific interface for the selected function.
Estimates	Heights Manager, STS	The creation of an invoice marked with a special flag as "estimate." The estimate may be referenced later to create an actual invoice.	The user selects the Estimate function.	An estimate is created and stored in the database.
Employee Services	Heights Manager, STS	A collection of functions for dealing with employee data. Entering, editing, and viewing data.	The user selects the desired function from the system interface.	The specific interface for the selected function.

Enter New Customer	Heights Manager, STS	The user creates a new customer record including name, address, and other information.	The user creates the New Customer function.	A new customer record is created in the database.
Create Invoice	Heights Manager, STS, Heights Employee	The user creates an invoice for a service performed.	The user selects the Invoice function.	An invoice is created, possibly printed out, and stored in the database.
Find Invoice	Heights Manager, STS, Heights Employee	The user searches for an invoice by selected criteria (date, invoice number, customer, etc.)	The user selects the Find Invoice function or the Customer History function.	A list of matching invoices is displayed.
Customer History	Heights Manager, STS	A work history for a particular customer, in the form of an invoice list.	The user selects the Customer History function.	A list of matching invoices is displayed.
Enter new Employee	Heights Manager, STS	The user creates a new employee record including name, address, and other information.	The user creates the New Employee function.	A new employee record is created in the database.
Reports	Heights Manager, STS, Accountant	Reports based on data for sales, tax, parts, etc. (see Functional Requirements) are created by STS.	The user selects the desired report function.	The desired report is created.
System Maintenance	System Management	A system manager will perform updates or repairs to the system as needed.	Desire for new functionality or problems with the system.	STS is upgraded or repaired.

Table 3.1 Use –Case Descriptions

### 3.6.2 Data Dictionary

NAME	DESCRIPTION	DATA TYPE
Date	Date of invoice	Date
Invoice Number	Invoice number	Auto number
Customer Name	First and last names	Text
Customer Address	Street, city, state, zip	Text
Customer Phone Number	Home, work, and cell numbers	Text
Vehicle Make	Vehicle make	Text
Vehicle Model	Vehicle model	Text
Vehicle Year	Vehicle year	Text
Engine	Engine displacement	Liters or cubic inches
License Plate	License plate of vehicle	Text
Mileage-In	Vehicle mileage when first brought in	Number
Mileage-Out	Vehicle mileage before leaving station	Number
VIN Number	Vehicle identification number	Text
Service(s)	line items of Service Description and Hours of Labor	Field
Service Description	Service description	Text
Hours of Labor	Hours of labor (per service)	Decimal
Labor total	total (Hours of Labor x Hourly Rate) [Hourly Rate currently = \$68]	\$
Parts	line items of Part Description, Part Quantity, Part Retail Price, and Warranty	Field
Part Description	Part description	Text
Part Quantity	Part quantity	Number
Part Retail Price	Part retail price	\$
Warranty	Warranty on part	Text
Parts Total	total (Part Quantity x Part Retail Price)	\$
Sublets	line items of Sublet Description and Sublet Price	Field
Sublet Description	Description of outside work performed	Text
Sublet Price	Price of sublet work	\$
Sublet Total	total (Price of Sublet Work)	\$
EPA	line items of Recycled Item and Recycling Cost	Field
Recycled Item	Recycled Item	Text
Recycling cost	cost for Recycled Item	\$
EPA Total	total (Recycling Cost)	\$
Subtotal	Labor Total + Parts Total + Sublet Total + EPA Total	\$

<b>NAME</b>	<b>DESCRIPTION</b>	<b>DATA TYPE</b>
Tax	(tax rate x Subtotal) [current tax rate = .06]	\$
Total	Subtotal + Tax	\$
Notes/Reminders	Area for typed notes or system-generated reminders	Field
Heights Warranty	Statement of general warranty: "3,000 miles or 90 days."	Text
Servicing Technician	Mechanic who performed the service(s)	Text
Parts Statement	Statement of general parts condition: "All parts new unless otherwise specified."	Text
Authorization Agreement	Statement of authorization with room for customer signature	Text
Company Header	Company information: Heights Service Station 70 Glenn Avenue Midland Park, New Jersey, 07432 444-6753	Text
Customer	Customer Name, Customer Address, Customer Phone Numbers, Comments	Record
Vehicle	Vehicle Make, Vehicle Model, Vehicle Year, Vehicle Color, License Plate, VIN, Engine, Customer	Record
Employee	Employee Name, Employee Address, Employee Phone Numbers, Employee SS#, Vacation Days, Deductions, Withholdings, Comments	Record
Warranty	Description, Duration, Mileage, Comment	Record
Supplier	Supplier Name, Contact Name, Contact Title, Supplier Address, Supplier Phone Numbers, Payment Terms, Comments	Record

Table 3.2 Data Dictionary

## 4. Architectural Design

### 4.1 System Structuring

#### 4.1.1 Repository Model

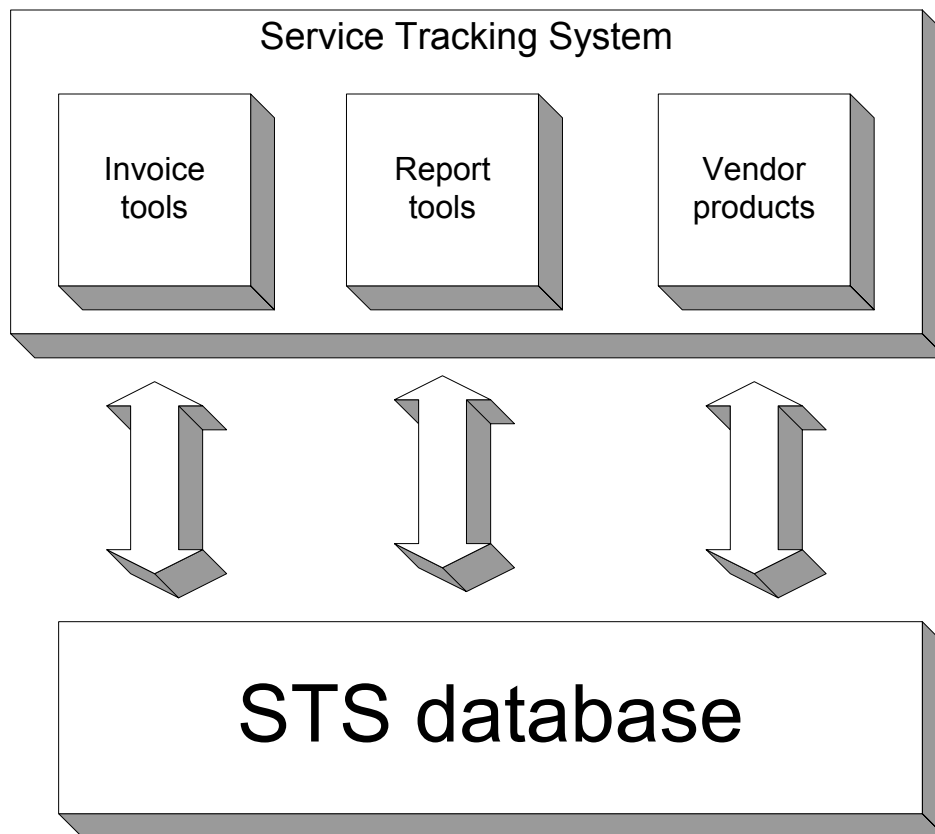


Figure 4.1 Repository Model Diagram

Service Tracking System will have a central database that is accessed by all other areas of the system. For this reason, a repository model will best represent the system structure. Sommerville, in the book *Software Engineering*, lists many design aspects that are modeled well by the repository model. In general, a repository model has "...all shared data...held in a central database that can be accessed by all subsystems." Furthermore, "this model is...suited to applications where data is generated by one sub-system and used by another." Service Tracking System will have all of its data stored in a central

database that will be accessed by all other functions of the system. As noted by Sommerville, there are several advantages of a shared repository structure:

- It is an efficient way to share large amounts of data since there is no need to transmit it explicitly from one sub-system to another.
- It is easy to integrate new tools into the system, for access and security control, or for added functionality.

Sommerville also notes some disadvantages:

- Performance and flexibility may be compromised because all sub-systems must agree on a common repository data model.
- Distribution and evolution of the repository may be difficult to impossible due to the shared data model.

We find these to be acceptable disadvantages considering the small nature of the program. We do not anticipate major upgrades to STS because the nature of the business is fairly static. Also, because our client is a privately owned company that will run STS on a single PC, the need for system distribution does not exist.

## 4.1.2 Abstract Machine Model

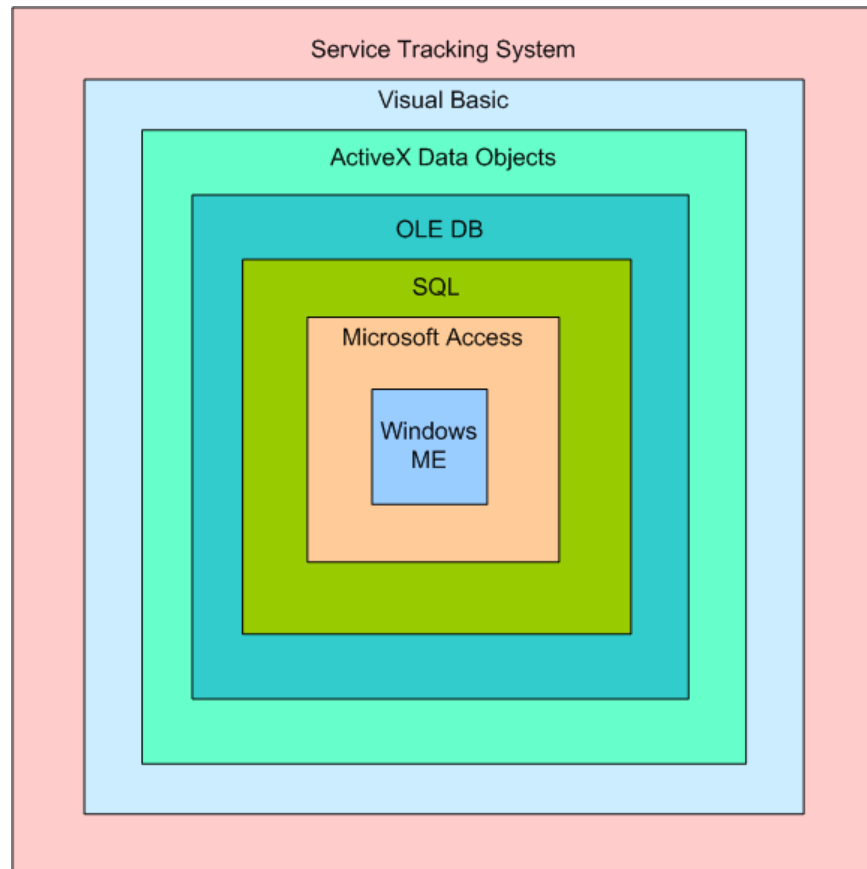


Figure 4.2 Abstract Machine Model

This abstract machine model represents the layers of functionality available to Service Tracking System. Through requirement gathering, it is known that STS will run in the **Windows ME** operating system. **Microsoft Access** is the DBMS and data store that will run on Windows. **SQL** is the OLE database provider. Its "... purpose is to grab the data from the data store and provide it to the requesting application. It will break down the native data into a stream that can be accessed using standard programmatic interfaces." **OLE DB** is a key component of universal data access. It "...is a set of COM-based

programming interfaces that allow transparent data access, but in such a way that the client applications don't know or need to know what kind of data they are connected to.”

By utilizing OLE DB, Service Tracking System becomes more flexible and powerful by being able to access any type of database (<http://www.raritanval.edu/departments/CIS/full-time/Schwarz/avb/Lesson3.htm>). The **ActiveX Data Object (ADO)** layer provides a higher level of control of OLE DB data than the OLE DB provides. ADO makes programming easier in the same way that a higher-level language such as C++ is easier when compared to a low level of control like machine language. Other advantages of using this ADO layer is that it is programming language neutral (works with VB, C++, Java, etc.), provider neutral (can access any OLE DB source), and provides all of the same OLE DB functionality ([http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnuda/html/msdn\\_dbado.asp](http://msdn.microsoft.com/library/default.asp?url=/library/enus/dnuda/html/msdn_dbado.asp)). Visual Basic will be used to create the actual windows as well as to implement the ADO layer.



### 4.1.3 System Organization Diagram

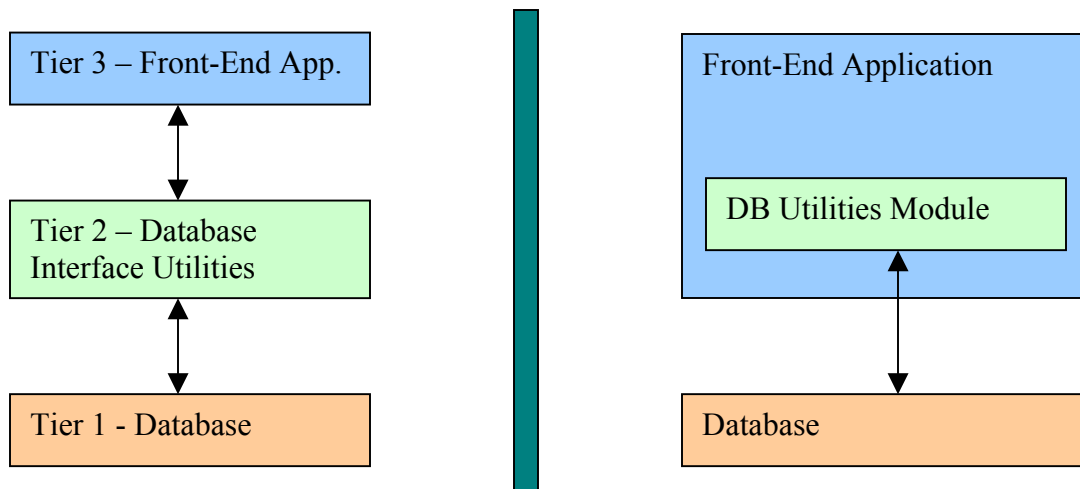
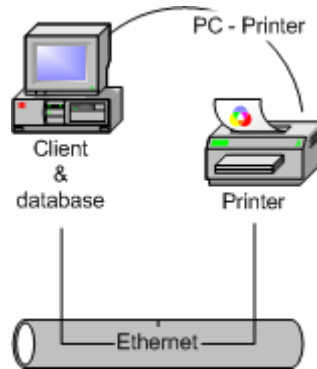


Figure 4.3 System Organization Diagram

The system's structure will be three-tiered and will be implemented in two actual components: a Microsoft Windows® front-end application, and a relational database for the back-end and information storage. The front-end will be tier three, with tier two (the database interface) built into it. Tier one will be the actual database itself. The left side of this diagram is the overall design of the system, and the right side is the implementation.

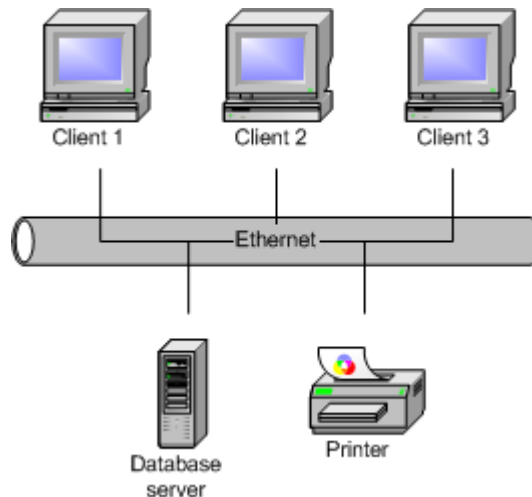
#### 4.1.4 Hardware Organization Model

##### TO-BE implementation:



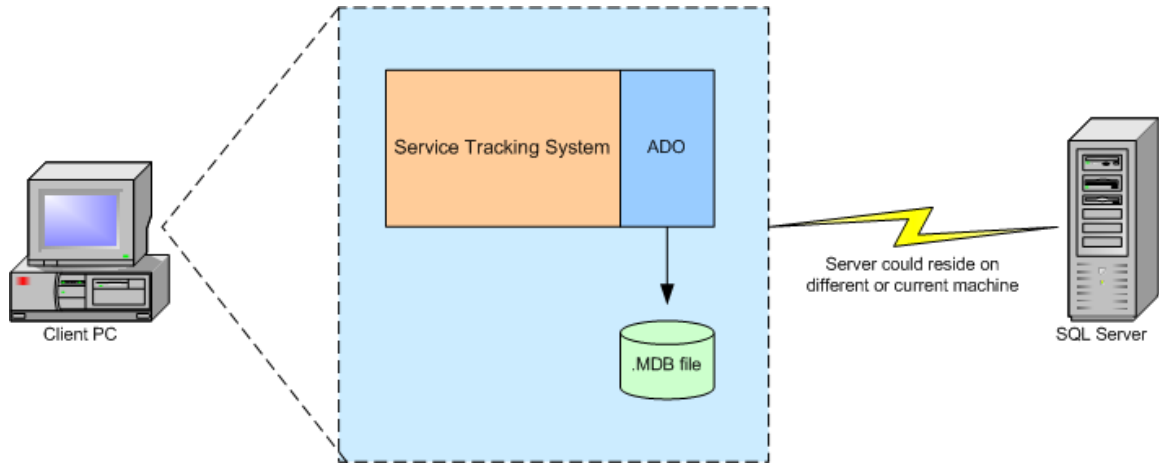
Service tracking system will be implemented on a single personal computer with access to a single printer.

##### Supported implementation:



Service Tracking System is designed to be scalable. If the needs of Heights Service Station change in the future, the system can be expanded by having the database server reside on a separate machine, while clients can run the applications from individual workstations.

## Underlying Scalability Support:



As outlined in the Abstract Machine Model above, the use of an ADO layer within the system structure allows for different types of databases to be utilized by Service Tracking System, as opposed to the current Microsoft Access database (.MDB). Also, the use of an SQL layer allows the SQL server to run off of a separate machine in a LAN setting. If Heights were to expand and wanted to access STS from different areas of the shop, this networked implementation would be valuable.

## 4.2 Control Models

A Call-Return Centralized Control Model will be used for Service Tracking System. The “main program” acts as the system controller and passes control to subsequent sub-routines. Consistent with this model, there is a tree of functions stemming from the main program and certain functions are only available from within other functions. For example, while *creating an invoice*, the system may transfer control to *entering a new customer* if a new customer name is entered into the name field of the invoice. When the new customer task is complete, control returns to the invoice.

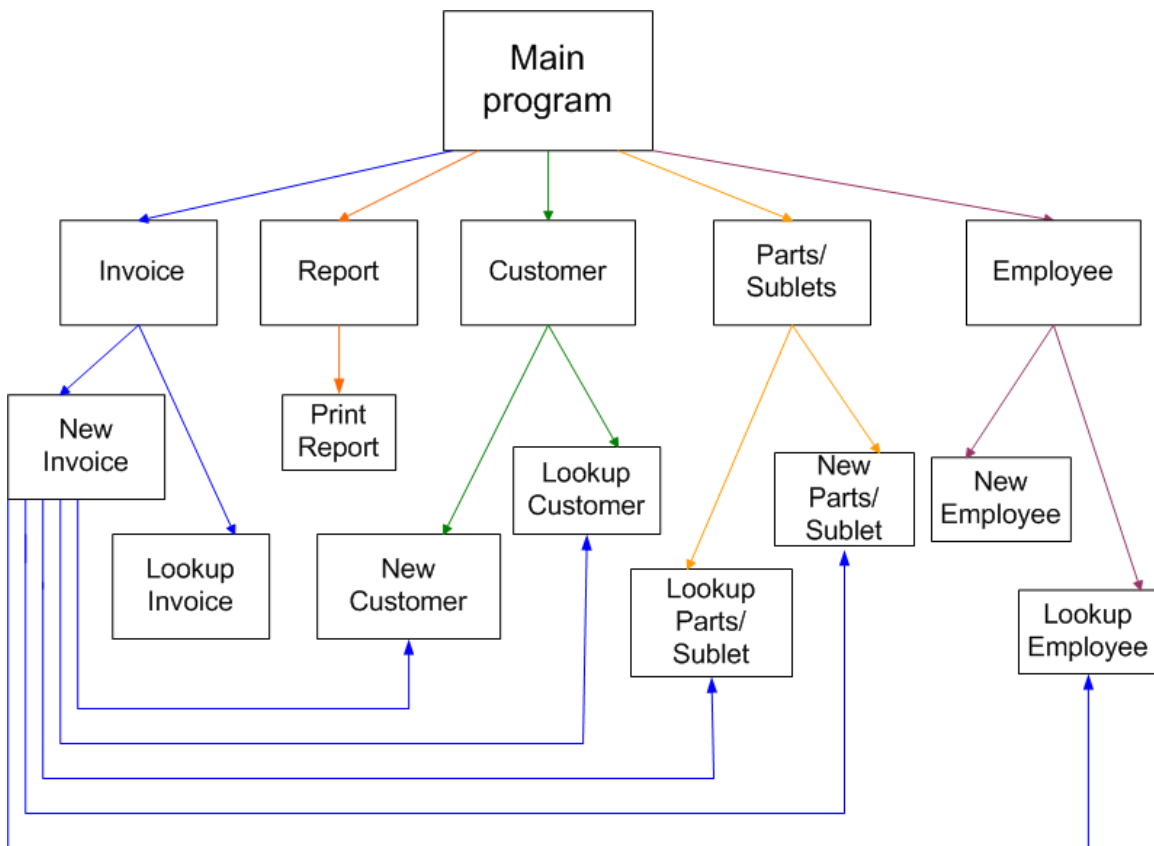


Figure 4.4 Control Model Diagram

## 4.2.1 Sequence Diagram

Sequence Diagram for Create Invoice:

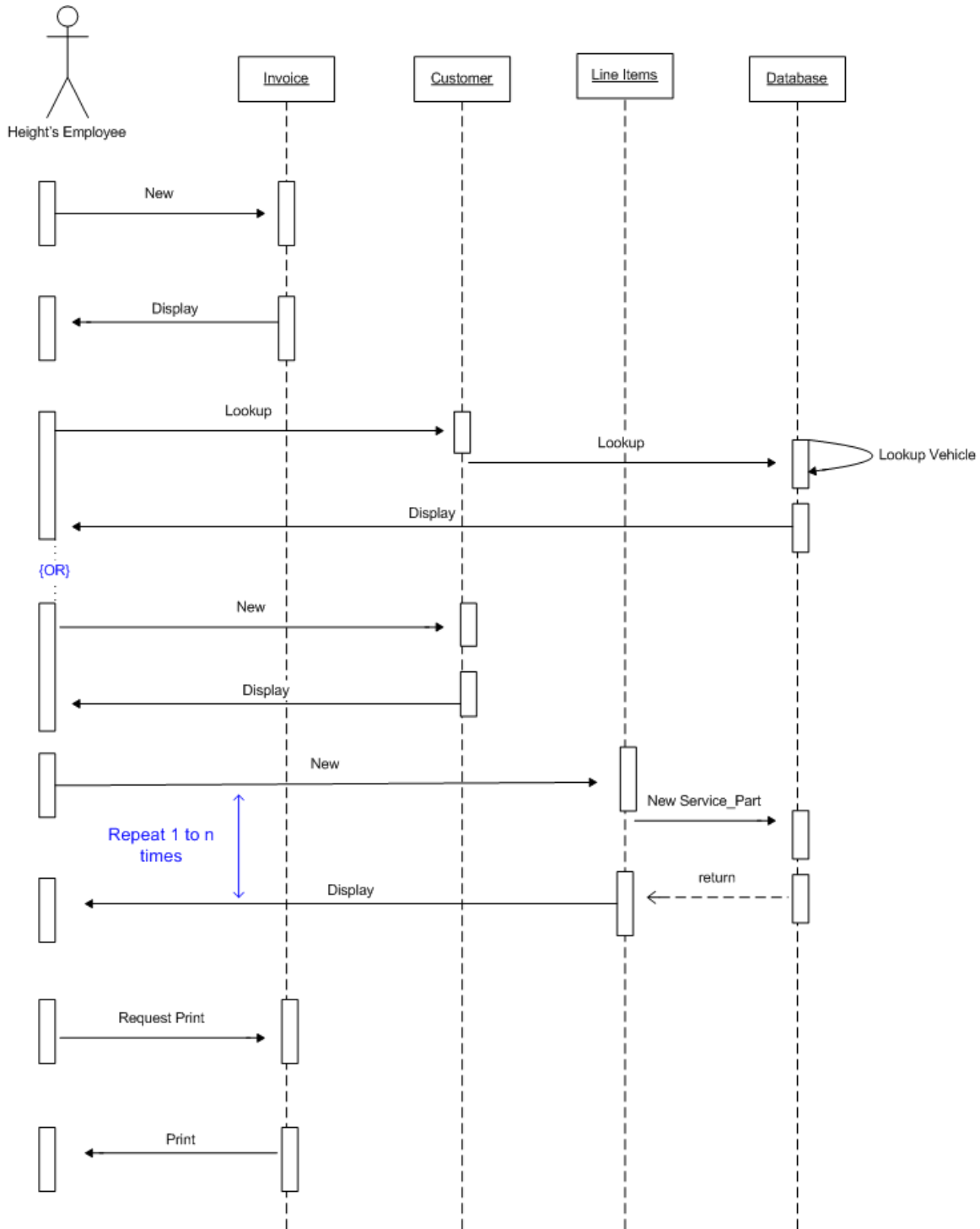
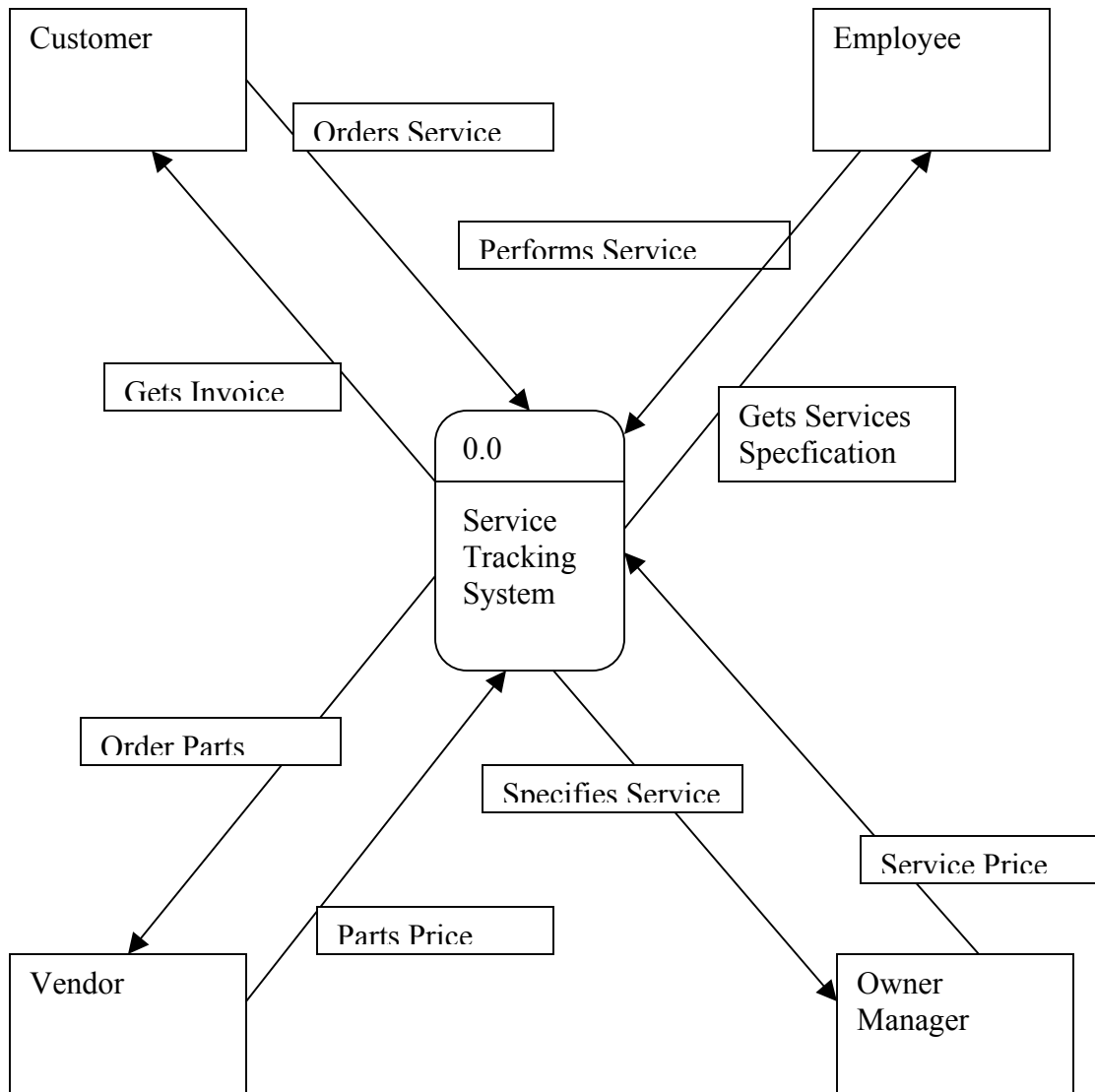


Figure 4.5 Sequence Diagram

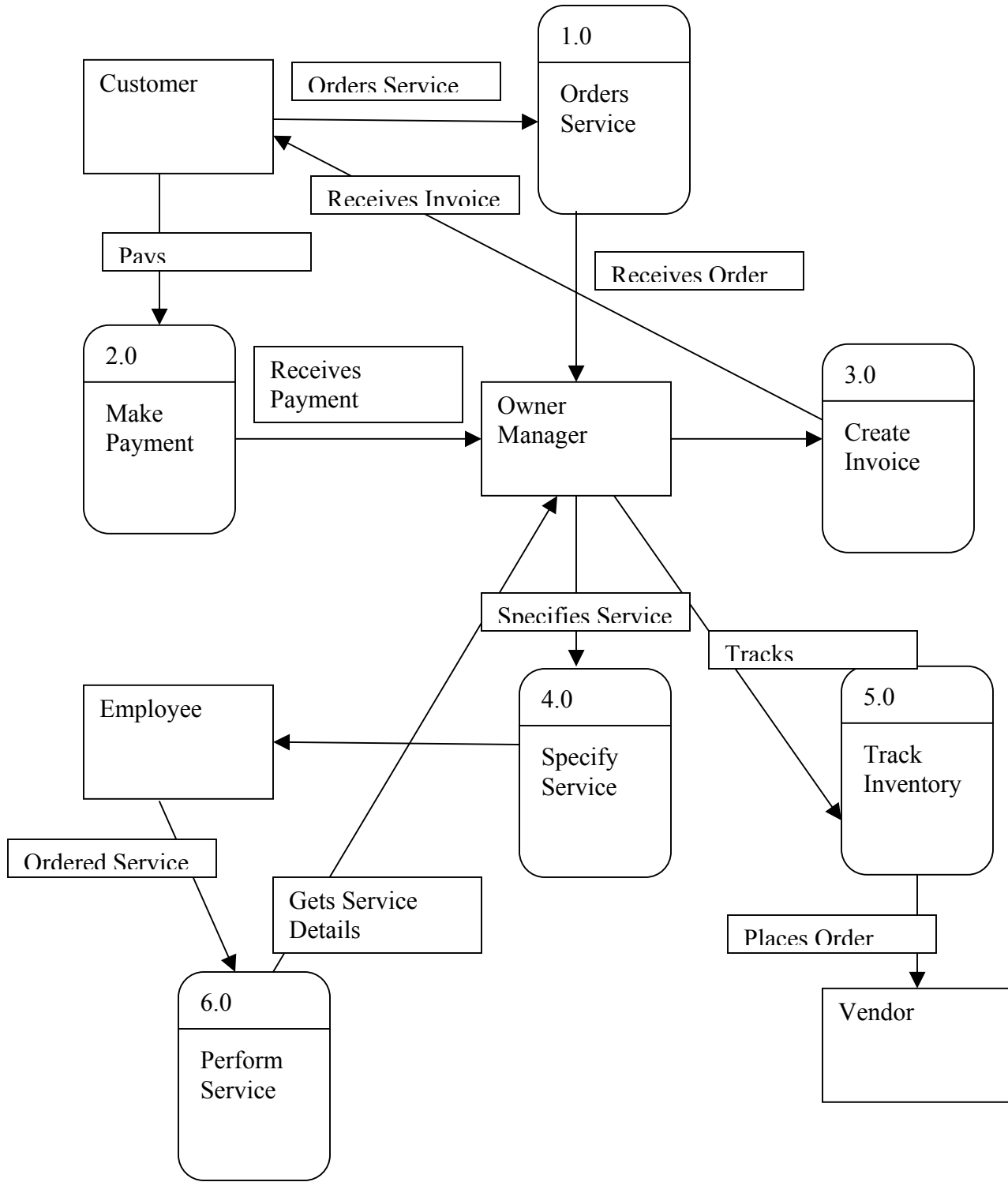
## 4.3 Modular Decomposition

### 4.3.1 Data Flow Diagrams

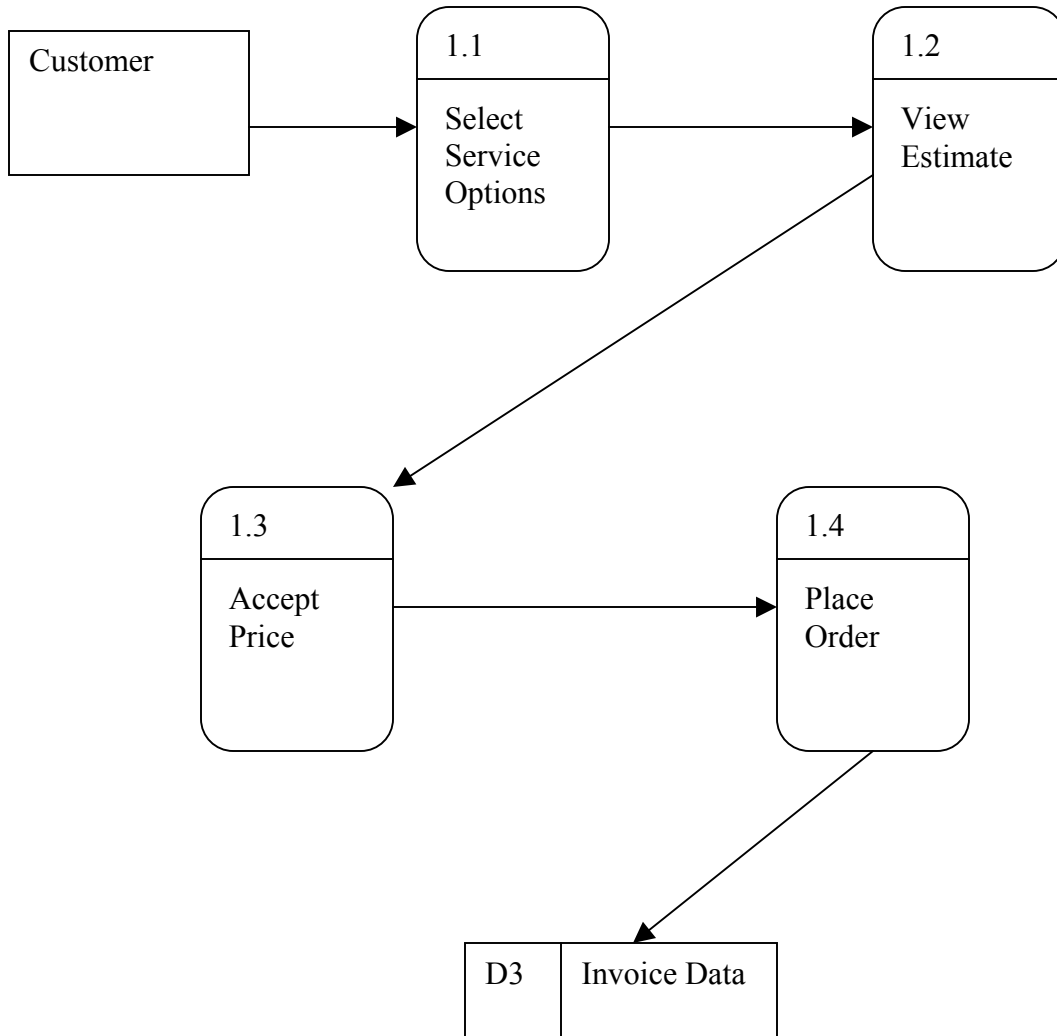
#### Context Diagram



# General Diagram

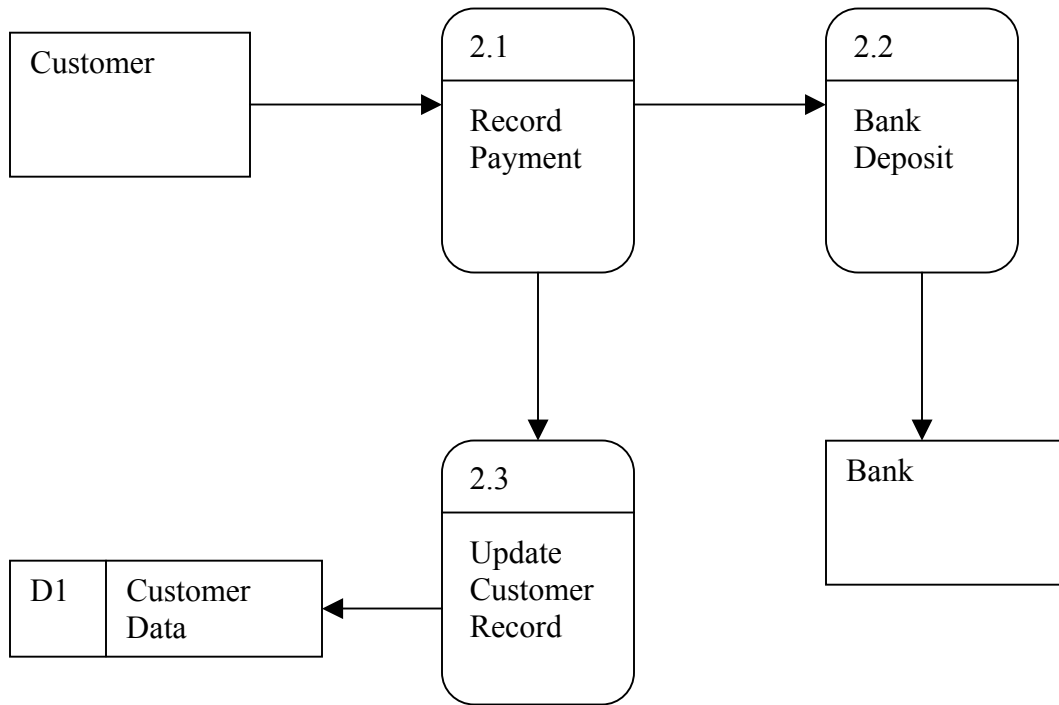


## Decomposition 1.0 (Order Service)

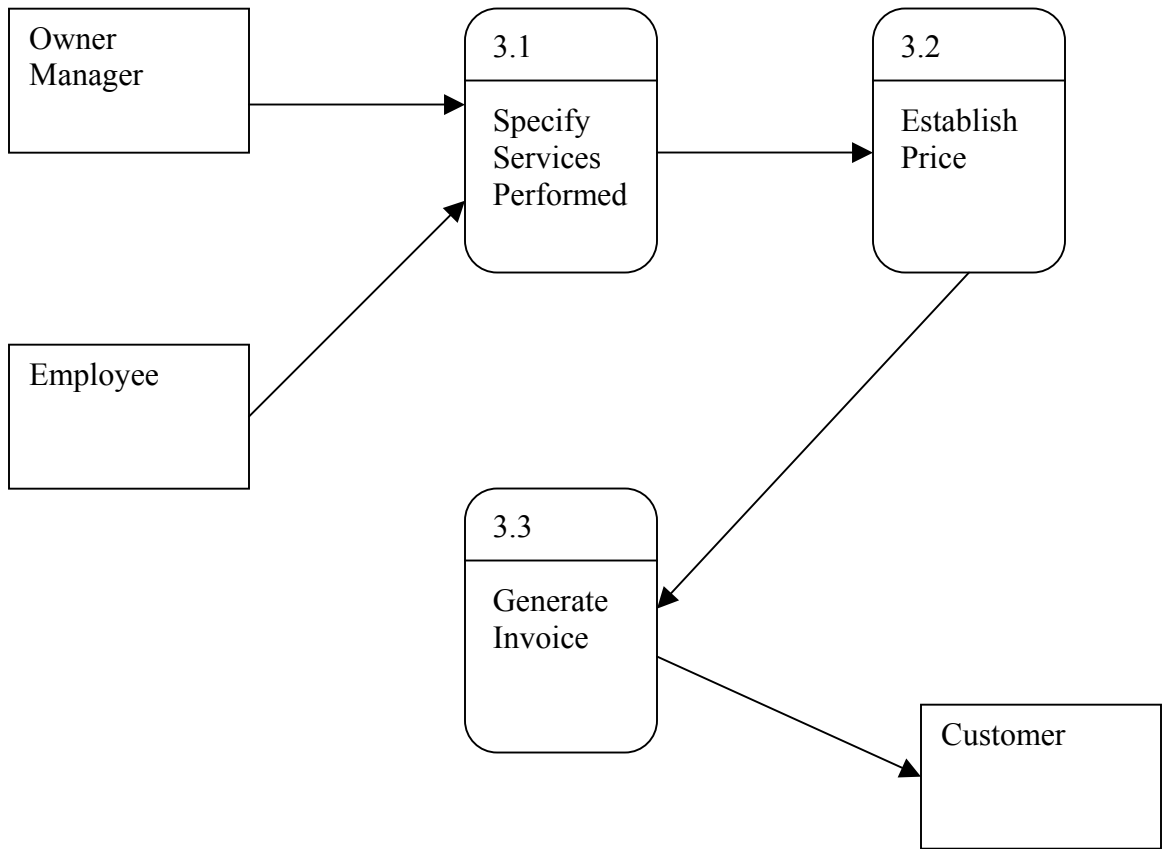




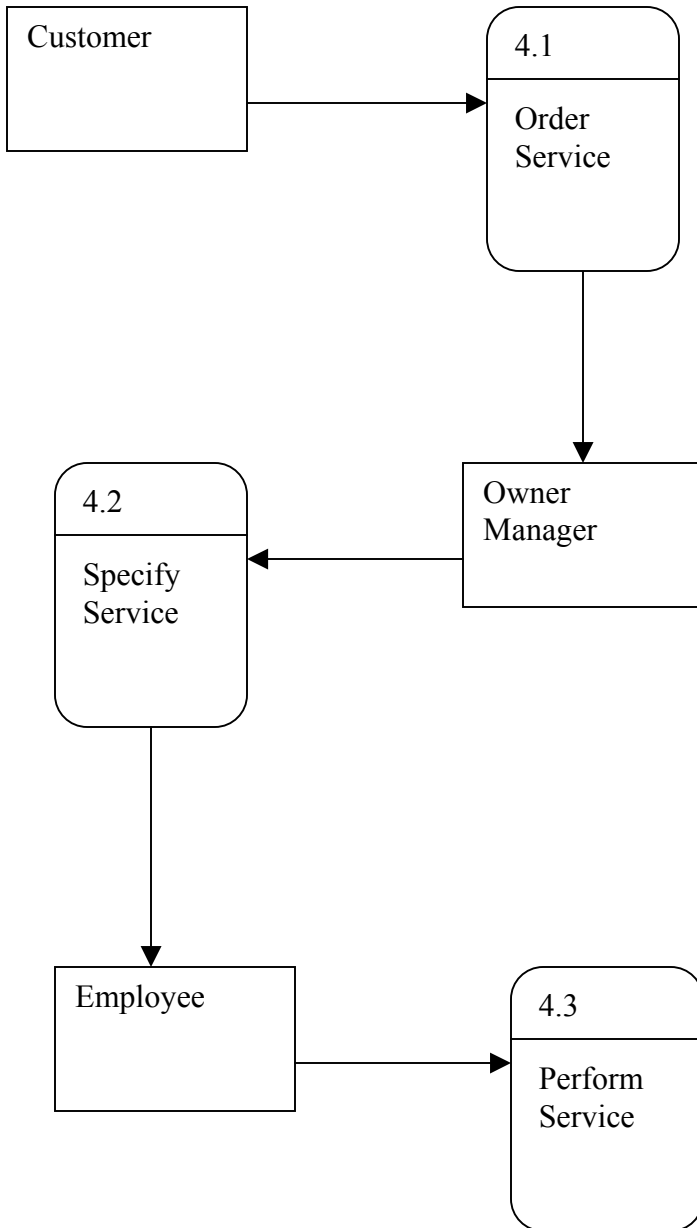
## Decomposition 2.0 (Payment)



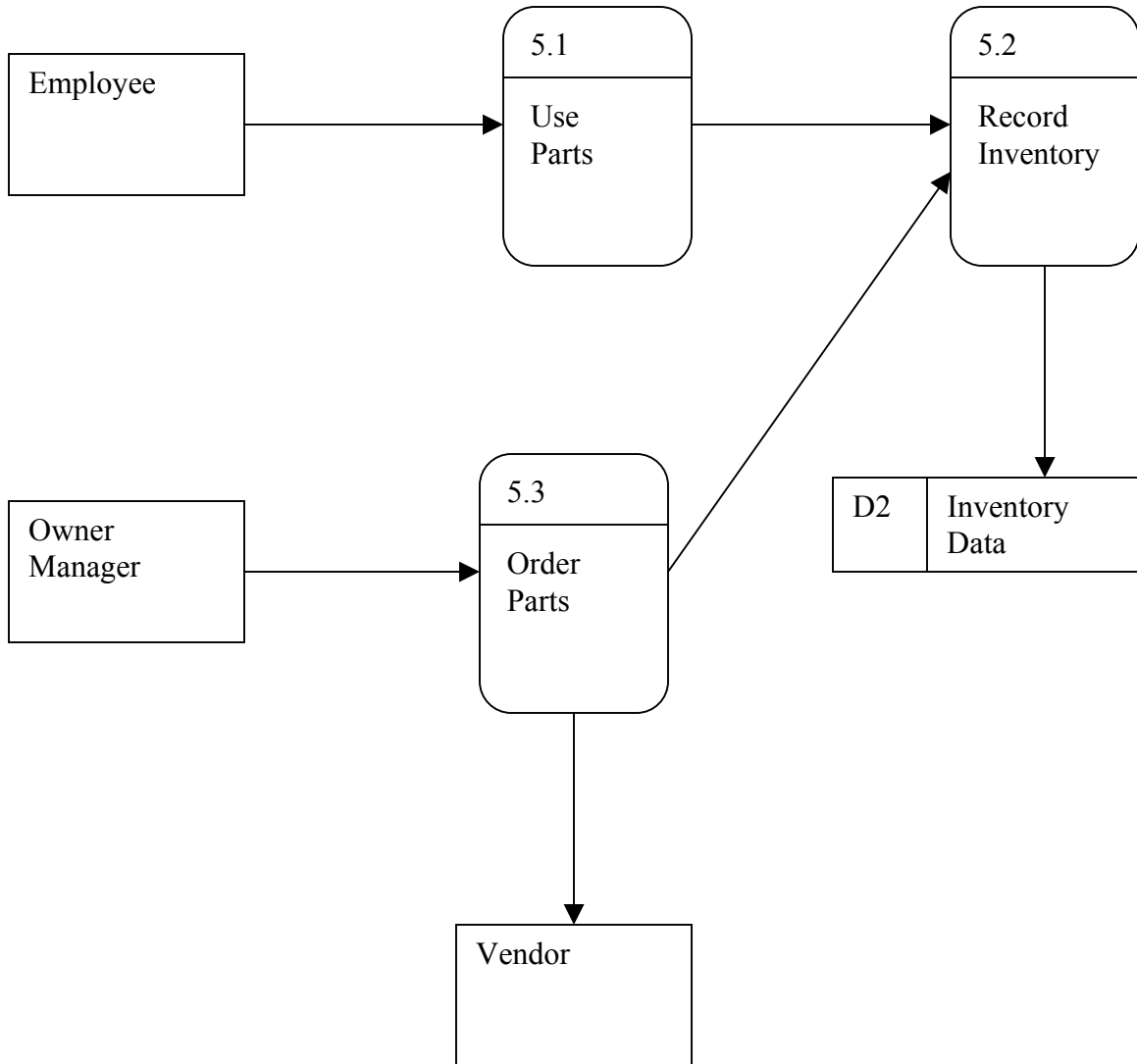
### Decomposition 3.0 (Invoice)



## Decomposition 4.0 (Service)



## Decomposition 5.0 (Inventory)



## 4.3.2 Object Model Diagram

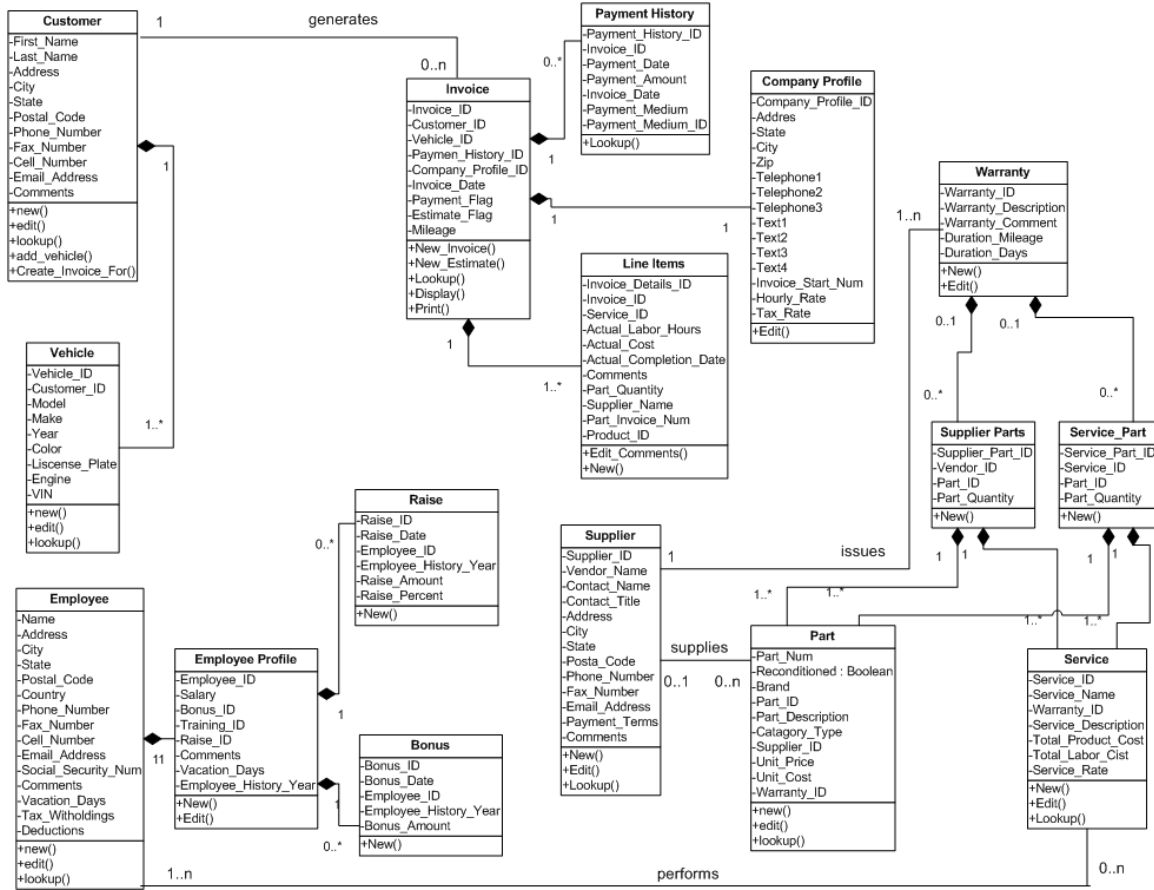


Figure 4.6 Object Model Diagram

The UML notation for this Object Model Diagram is as follows:

- Object aggregation is represented by a diamond-ended link. The object at the diamond end of the link is “composed of” the object(s) at the non-diamond end.
- For each object, the object name is in the top third of the rectangle, the object’s attributes are in the middle third, and the object’s operations are in the bottom third.

## 4.4 Database Design

### 4.4.1 Entity-Relationship Model

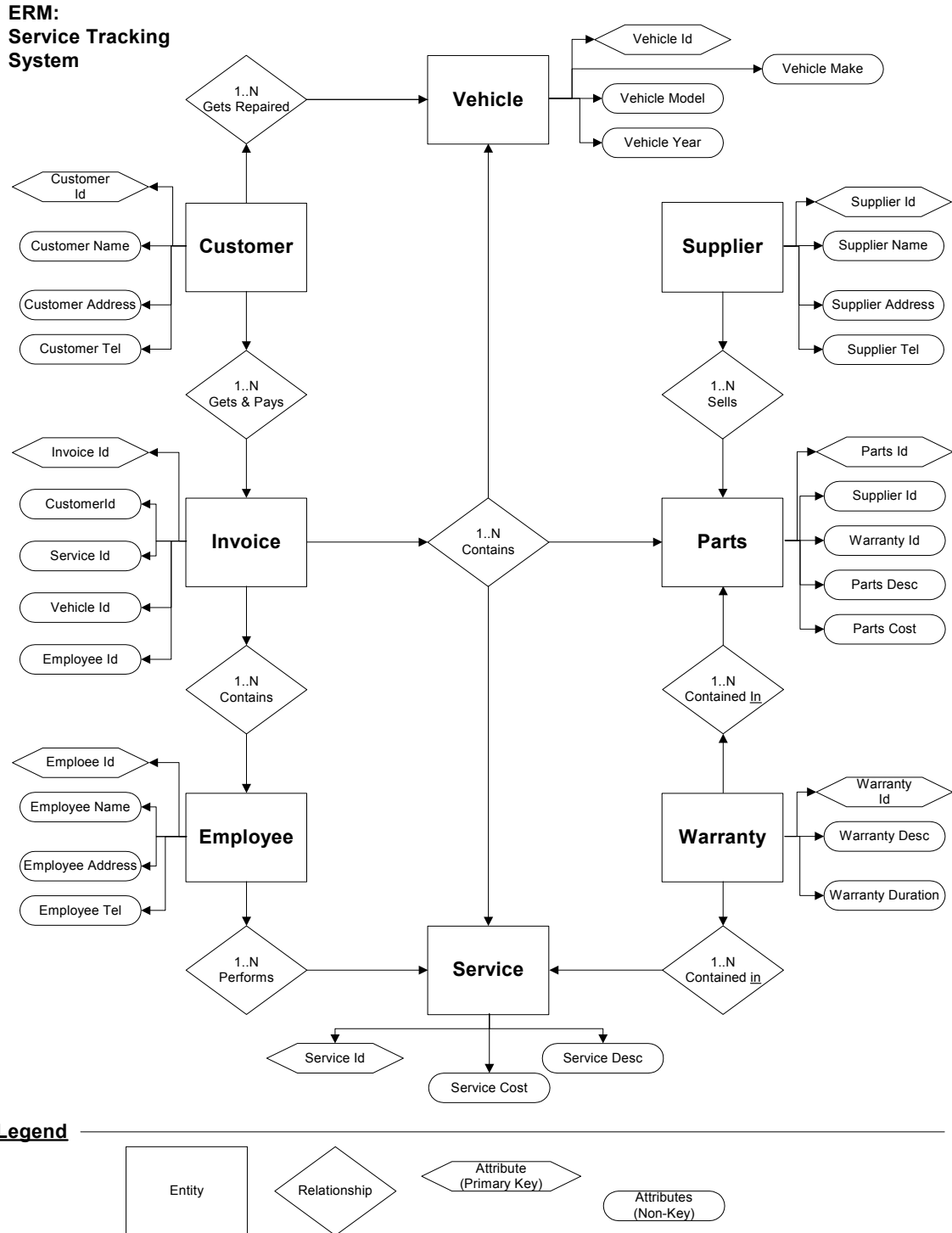


Figure 4.7 Entity-Relationship Model

## 4.4.2 Entity-Relationship Diagram

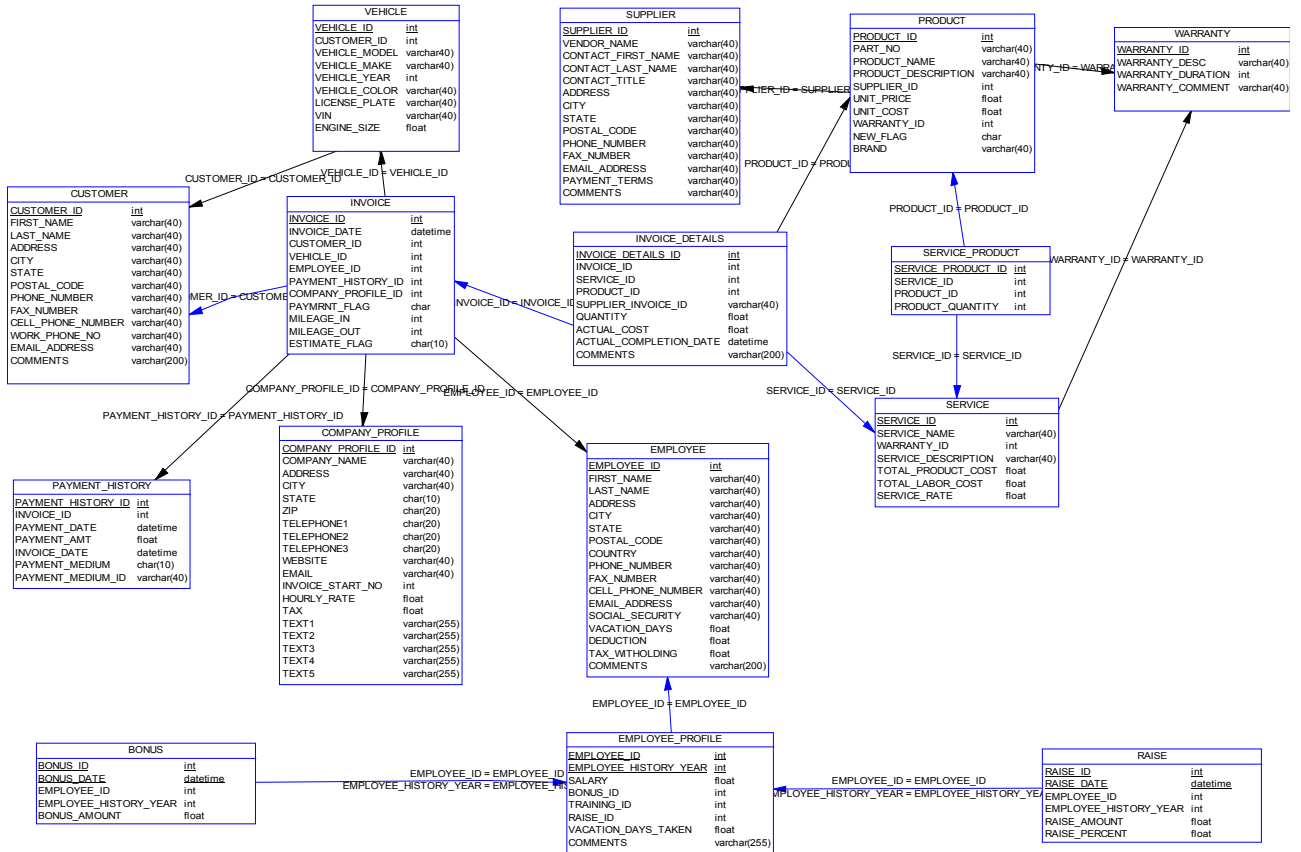


Figure 4.8 Entity-Relationship Diagram

### 4.4.3 Database Objects List

#### Table list

Name	Code
Customer	CUSTOMER
Supplier	SUPPLIER
Product	PRODUCT
Service	SERVICE
Service_Product	SERVICE_PRODUCT
Invoice	INVOICE
Invoice_Details	INVOICE_DETAILS
Employee	EMPLOYEE
employee_profile	EMPLOYEE_PROFILE
Bonus	BONUS
Raise	RAISE
VEHICLE	VEHICLE
Warranty	WARRANTY
company_profile	COMPANY_PROFILE
Payment_history	PAYMENT_HISTORY

#### Table columns list

Name	Code
customer_id	CUSTOMER_ID
first_name	FIRST_NAME
last_name	LAST_NAME
address	ADDRESS
city	CITY
state	STATE
postal_code	POSTAL_CODE
phone_number	PHONE_NUMBER
fax_number	FAX_NUMBER
cell_phone_number	CELL_PHONE_NUMBER



Name	Code
work_phone_no	WORK_PHONE_NO
email_address	EMAIL_ADDRESS
comments	COMMENTS
supplier_id	SUPPLIER_ID
vendor_name	VENDOR_NAME
contact_first_name	CONTACT_FIRST_NAME
contact_last_name	CONTACT_LAST_NAME
contact_title	CONTACT_TITLE
address	ADDRESS
city	CITY
state	STATE
postal_code	POSTAL_CODE
phone_number	PHONE_NUMBER
fax_number	FAX_NUMBER
email_address	EMAIL_ADDRESS
payment_terms	PAYMENT_TERMS
comments	COMMENTS
product_id	PRODUCT_ID
part_no	PART_NO
product_name	PRODUCT_NAME
product_description	PRODUCT_DESCRIPTION
supplier_id	SUPPLIER_ID
unit_price	UNIT_PRICE
unit_cost	UNIT_COST
warranty_id	WARRANTY_ID
new_flag	NEW_FLAG
brand	BRAND
service_id	SERVICE_ID
service_name	SERVICE_NAME
warranty_id	WARRANTY_ID

Name	Code
service_description	SERVICE_DESCRIPTION
total_product_cost	TOTAL_PRODUCT_COST
total_labor_cost	TOTAL_LABOR_COST
service_rate	SERVICE_RATE
service_product_id	SERVICE_PRODUCT_ID
service_id	SERVICE_ID
product_id	PRODUCT_ID
product_quantity	PRODUCT_QUANTITY
invoice_id	INVOICE_ID
invoice_date	INVOICE_DATE
customer_id	CUSTOMER_ID
vehicle_id	VEHICLE_ID
employee_id	EMPLOYEE_ID
payment_history_id	PAYMENT_HISTORY_ID
company_profile_id	COMPANY_PROFILE_ID
paymrnt_flag	PAYMRNT_FLAG
mileage_in	MILEAGE_IN
mileage_out	MILEAGE_OUT
estimate_flag	ESTIMATE_FLAG
invoice_details_id	INVOICE_DETAILS_ID
invoice_id	INVOICE_ID
service_id	SERVICE_ID
product_id	PRODUCT_ID
supplier_invoice_id	SUPPLIER_INVOICE_ID
quantity	QUANTITY
actual_cost	ACTUAL_COST
actual_completion_date	ACTUAL_COMPLETION_DATE
comments	COMMENTS
employee_id	EMPLOYEE_ID
first_name	FIRST_NAME

Name	Code
last_name	LAST_NAME
address	ADDRESS
city	CITY
state	STATE
postal_code	POSTAL_CODE
country	COUNTRY
phone_number	PHONE_NUMBER
fax_number	FAX_NUMBER
cell_phone_number	CELL_PHONE_NUMBER
email_address	EMAIL_ADDRESS
social_security	SOCIAL_SECURITY
vacation_days	VACATION_DAYS
deduction	DEDUCTION
tax_withholding	TAX_WITHHOLDING
comments	COMMENTS
employee_id	EMPLOYEE_ID
employee_history_year	EMPLOYEE_HISTORY_YEAR
salary	SALARY
bonus_id	BONUS_ID
training_id	TRAINING_ID
raise_id	RAISE_ID
vacation_days_taken	VACATION_DAYS_TAKEN
comments	COMMENTS
bonus_id	BONUS_ID
bonus_date	BONUS_DATE
employee_id	EMPLOYEE_ID
employee_history_year	EMPLOYEE_HISTORY_YEAR
bonus_amount	BONUS_AMOUNT
raise_id	RAISE_ID
raise_date	RAISE_DATE

Name	Code
employee_id	EMPLOYEE_ID
employee_history_year	EMPLOYEE_HISTORY_YEAR
raise_amount	RAISE_AMOUNT
raise_percent	RAISE_PERCENT
vehicle_id	VEHICLE_ID
customer_id	CUSTOMER_ID
vehicle_model	VEHICLE_MODEL
vehicle_make	VEHICLE_MAKE
vehicle_year	VEHICLE_YEAR
vehicle_color	VEHICLE_COLOR
license_plate	LICENSE_PLATE
vin	VIN
engine_size	ENGINE_SIZE
warranty_id	WARRANTY_ID
warranty_desc	WARRANTY_DESC
warranty_duration	WARRANTY_DURATION
warranty_comment	WARRANTY_COMMENT
company_profile_id	COMPANY_PROFILE_ID
company_name	COMPANY_NAME
address	ADDRESS
city	CITY
state	STATE
zip	ZIP
telephone1	TELEPHONE1
telephone2	TELEPHONE2
telephone3	TELEPHONE3
website	WEBSITE
email	EMAIL
invoice_start_no	INVOICE_START_NO
hourly_rate	HOURLY_RATE

Name	Code
tax	TAX
text1	TEXT1
text2	TEXT2
text3	TEXT3
text4	TEXT4
text5	TEXT5
payment_history_id	PAYMENT_HISTORY_ID
invoice_id	INVOICE_ID
payment_date	PAYMENT_DATE
payment_amt	PAYMENT_AMT
invoice_date	INVOICE_DATE
payment_medium	PAYMENT_MEDIUM
payment_medium_id	PAYMENT_MEDIUM_ID

Table 4.1 Database Object List

## **Appendix A Requirements Gathering Questionnaires**

A series of two questionnaires were given to the sponsor, Heights Service Station, at the early stages of the Requirements Gathering phase of this project. The responses to these questionnaires helped form the Functional Requirements as well as guided Service Tracking System in a direction suitable to the sponsor.

### **A.1 Questionnaire 1**

September 28, 2002

Mr. Sonderfan  
Heights Service Station  
70 Glenn Avenue  
Midland Park, NJ 07432

Dear Mr. Sonderfan:

The attached questionnaire is provided to give the CIS 491 project team some preliminary ideas as to the design of your software system.

Please take some time to answer the questions in as much detail as you can provide. Also feel free to elaborate or provide any additional information. Remember, the better we understand your needs, the better we can make your program!

Within the next few weeks, as system development progresses, some follow-up questionnaires may be required. If you would like to contact me before then with any questions, concerns, or comments, feel free to email me at [mvd0503@njit.edu](mailto:mvd0503@njit.edu) or call me at (973)-769-0819.

Thank you,

Marc DeFilippo  
New Jersey Institute of Technology  
CIS 491 Senior Project team member

1. What types of information do you provide on customer invoices?  
(for example: labor rate, mileage in/out, customer name, address, etc.)
  
2. What types of reports would be useful to your business?  
(for example: monthly billing, employee hours, number of jobs completed, etc.)
  
3. What information do you wish to store about your customers?  
(for example: name, address, history of work performed, referrals, etc.)
  
4. What information do you wish to store about your employees?  
(for example: address, phone number, hours, pay rate, certifications, etc.)
  
5. Do you wish to store contact information?  
(for example: phone number and names for trans shops, parts suppliers, etc.)
  
6. Personally, how do you refer to each job you perform? By customer name, the type of car, a customer number, or something else?
  
7. What types of payment do you accept?

## A.2 Questionnaire 2 Response

2/1) First + last names, date, Invoice #, P.O. #  
address; home, work, + cell ph. numbers  
car ~~year~~ make + model, eng. displacement (l or cid.)  
lic. plate, mileage, vin #

- Description of ~~car~~ service provided, time spent  
times by rate per service, labor total
- Itemized part list (by type, not brand), quantity +  
customer price. parts total
- Sublet items, alignment, transmission rebuild  
~~env~~ environmental recycling fee: tires, oil, filters  
antifreeze

Subtotal, tax, Total. Type of payment

Notes for future work needed

for internal use: Brand + part # + supplier invoice # for  
recondition parts used + warranty terms

- possible an asterisk (\*) or other symbol to reference footnote  
for 90 days, 1 yr, lifetime warranty or "other"
- a statement of general warranty on <sup>parts</sup> labor - 9,000 miles, 90 days  
extended parts warranty as noted by \* or other symbol
- technician who worked on car
- statement "All parts new unless otherwise specified"
- signature of customer for authorization + agreement to limited  
liability statement

1



#2 - parts sales by employee  
~~labor sales total~~ labor sales total, parts sales total  
labor hours billed per employee  
Sales tax total  
Sublet type + total  
# of each type of service.

#3  
Name, address, ph #s, car yr make + model  
history of work performed  
work recommendations be done  
~~etc~~

4) Name, address, ph #, deductions claimed  
Tax withholding, all pay roll into  
vacation days provided and taken.  
Excellent work performed  
Screw ups or "come backs"

5) parts supplier list: Name, ph #, part or service

6) usually by customer name + then by service performed  
and when.

7) cash, check, mstr + VISA cards

I would need to be able to create an estimate  
& then from the estimate, a bill

I would need to generate maintenance needed reminders  
to be mailed & track who responds & to what service.

## A.3 Questionnaire 2

October 3, 2002

Mr. Sonderfan  
Heights Service Station  
70 Glenn Avenue  
Midland Park, NJ 07432

Dear Mr. Sonderfan:

Many thanks for your valuable response to our first questionnaire. Attached are some additional questions that will further enhance and fine-tune our understanding of your needs and wants.

Again, we greatly appreciate the time you spend providing detailed answers. If you would like to contact me with any questions, concerns, or comments, feel free to email me at [mvd0503@njit.edu](mailto:mvd0503@njit.edu) or call me at (973)-769-0819.

Thank you,

Marc DeFilippo  
New Jersey Institute of Technology  
CIS 491 Senior Project team member

1. Do you accept partial payments for work performed, and if so, how do you currently deal with that situation?  
(For example, do you make a note on the bill? Write a new bill?)
  
2.
  - a. For which types of services would you like the system to produce reminder mailings?
  
  - b. The system can generate automatic reminders in a number of ways. It can go by the time since last service (like 4 months after an oil change, produce a reminder for that customer) and it can go by the actual mileage of a vehicle (For instance, when a customer comes in for work, the vehicle's mileage is entered onto the bill. The system can check this mileage against the vehicle's work history and know that it's been 10,000 miles since the last tune-up and will produce a reminder.  
Would you like these mileage-based reminders to be printed directly on the bill, or should they be printed separately for mailing just like the time-based reminders?
  
3. Are any of your services a set price (like \$24.99 for any oil change)? If so please list them with their prices.
  
4. Is hourly rate the same for all services? What is your hourly rate?
  
5. Please outline how your current system works (i.e. hand written bills, filing cabinets, report generating...)
  
6. Please explain why this new computer based system will be valuable to you. What is wrong with the current system?

## A.4 Questionnaire 2 Response

10/8

HEIGHTS SERVICE STATION, INC.  
70 GLEN AVENUE  
MIDLAND PARK, N.J. 07432

#1) - 99% of work performed is paid when car is picked up.  
- Sometimes there are multiple payments. Then I retain all copies of the bill 'til paid in full.  
- On extreme cases of multiple payments I have a small generic invoice that I record payments and additional work as a running balance.

#2) - oil changes / 3mo - 3,000mi. Tune ups + trans service / 30,000 mi  
Tire rotation / 6,000mi, most tuning jobs / 60,000mi  
coolant flush / 24mo. Fuel injection cleaning / 30,000 mi

Mileage based reminders are most useful except for cooling system service.

I prefer ~~separate~~ separate email reminders

There are also seasonal triggered reminders such as:

- Winter is coming soon, to insure the best heat - have your thermostat repaired + anti-freeze checked for winter
- Summer is coming soon. Is your AC up to it?
- Now that Spring is here and your windows are open, have those new noises checked out. They are your car talking to you. . . . Just some examples of a seasonal reminder that could appear on a bill or on a routine reminder (oil change) depending on the season.

#3) I don't currently run specials. But it would be great to be able to generate business during our seasonally slower periods.

- MIDLAND PARK, N.J. 07633  
TO GREEN AVENUE  
HEIGHTS SERVICE STATION, INC.
- #4) Current hourly rate is \$68<sup>00</sup>/<sub>hr.</sub> & all jobs are billed at that rate.
- #5) - Currently estimates are verbal or just noted on a pad of paper. Bills are written by hand (like your sample) Bills are triplicate: customer gets top copy, yellow copy gets saved by month for sales tax records. Hard copy gets filed alphabetically by year. I have file space for 6 yrs in my office. I use sales tax figures for gross income reporting. (Profits and Success come from God's good hands). My accountant does a year-end report.
- #6) The computer based system would be a big help when customers bring in their car & say "do what ever it needs". This depends a lot on what has been done, when, in the past I take time to dig through the filing cabinets to reconstruct the repair/service history. Sometimes it is important to look up a job for warranty purposes. Sometimes I need to duplicate an invoice at a later date.
- It all boils down to time - the less time I spend behind the desk, the more time I can be productive in the shop ... and the sooner I could get home ... 13 to 15 hour days have been normal for a long time.

## Appendix B

### Prototype Evaluation

Service Tracking System went through six revisions before a final version was completed. Following are two surveys evaluating the last prototype of STS. One survey was for this team to complete, and the other was for the sponsor. Results of both surveys had a great influence on the final changes and additions to Service Tracking System.

#### B.1 Team Evaluation Results

##### Statement

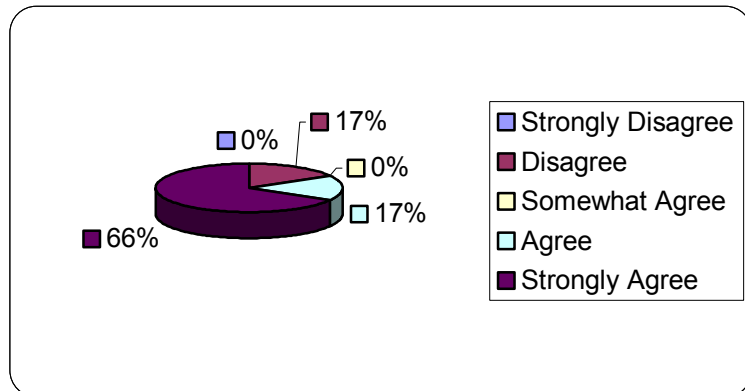
The colors used make the program easy to see and use.

##### Number of Responses

Strongly Disagree: 0  
Disagree: 1  
Somewhat Agree: 0  
Agree: 1  
Strongly Agree: 4

##### Agreement Rating

4.3



##### Comments

“Variety of colors could be used to make it more user friendly.”

**Statement**

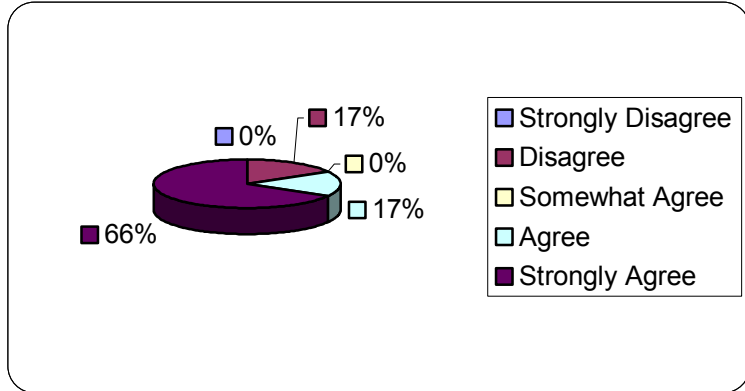
The fonts used make the program easy to see and use.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 1  
Somewhat Agree: 0  
Agree: 1  
Strongly Agree: 4

**Agreement Rating**

4.3



**Comments**

“Different fonts could be used to distinguish between the features (e.g. Bold letters on the buttons). This would make it easier to read.”

**Statement**

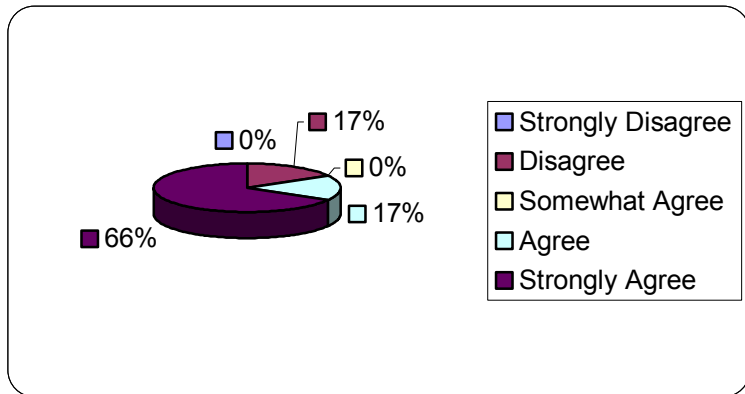
Button and field labels are easy to read.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 1  
Somewhat Agree: 0  
Agree: 1  
Strongly Agree: 4

**Agreement Rating**

4.3





**Statement**

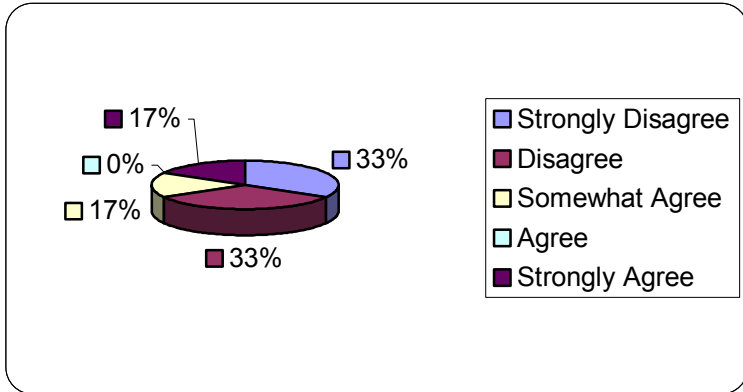
Screens contain too much information; they are difficult to look at.

**Number of Responses**

Strongly Disagree: 2  
Disagree: 2  
Somewhat Agree: 1  
Agree: 0  
Strongly Agree: 1

**Agreement Rating**

2.3



**Statement**

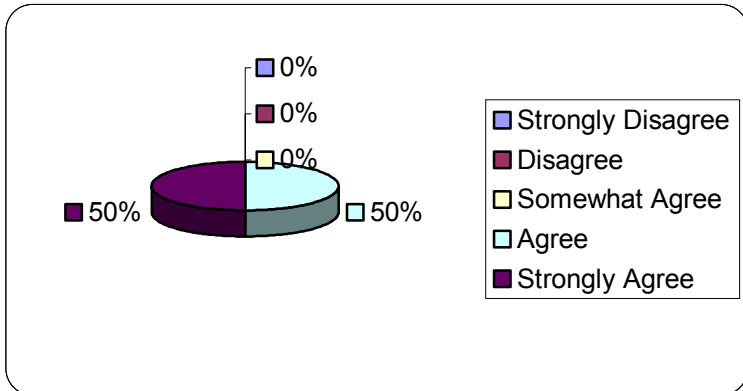
All of the most common tasks are easy to get to from the opening screen.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 0  
Agree: 3  
Strongly Agree: 3

**Agreement Rating**

4.5



**Statement**

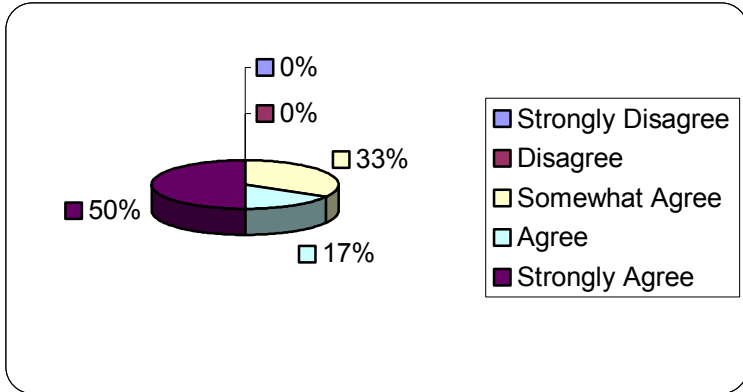
All of the most common tasks are easy to get to from most any screen I am in.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 2  
Agree: 1  
Strongly Agree: 3

**Agreement Rating**

4.2



**Statement**

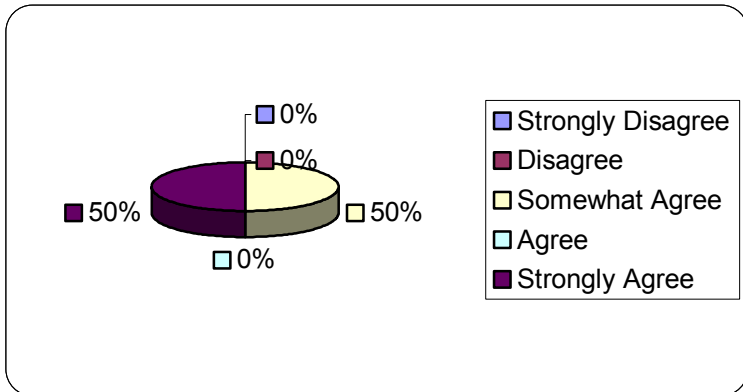
The items in the top menu bar belong in the menu they are in.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 3  
Agree: 0  
Strongly Agree: 3

**Agreement Rating**

4



**Comments**

“In the main screen, in the *Main* menu, perhaps *New* should be its own separate menu.”

“Items in the menu bar may need some reorganization. Perhaps we will let the team debate this. Perhaps even once again debate the main screen layout.”

**Statement**

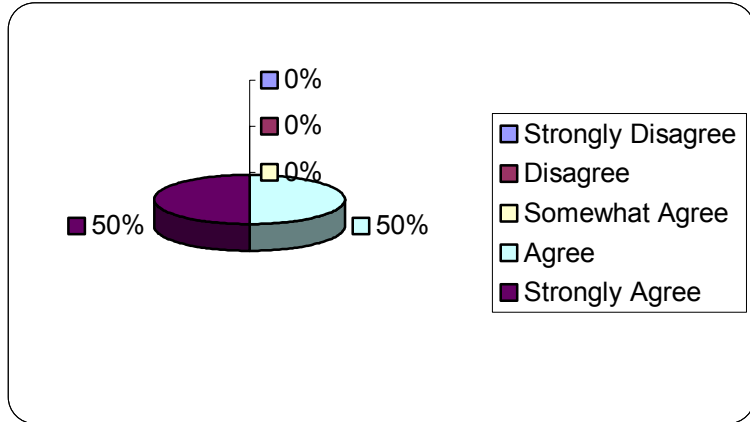
I can accurately predict what will happen for each button I click.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 0  
Agree: 3  
Strongly Agree: 3

**Agreement Rating**

4.5



**Statement**

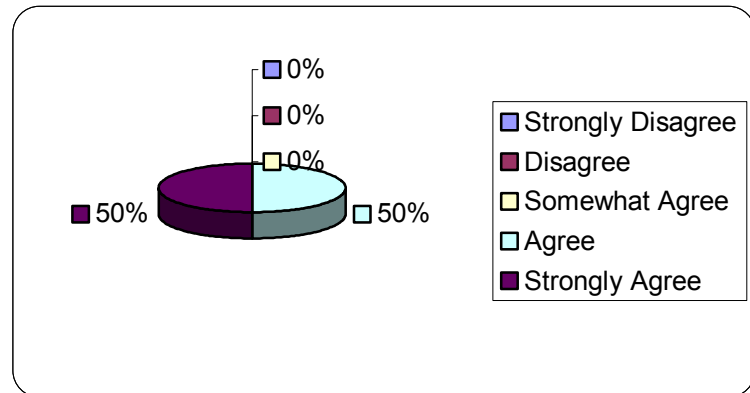
It is easy to navigate through all the tasks needed to be performed.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 0  
Agree: 3  
Strongly Agree: 3

**Agreement Rating**

4.5



**Statement**

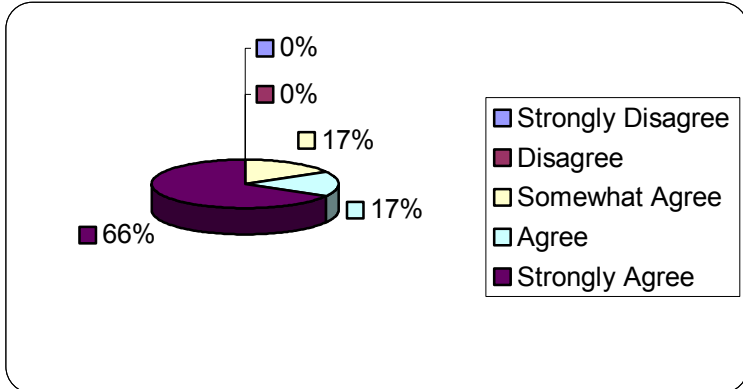
All of the tasks requested by the client are included in the program.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 1  
Agree: 1  
Strongly Agree: 4

**Agreement Rating**

4.5



**Comments**

“No visible function for Reminders/Mailings.”

**Statement**

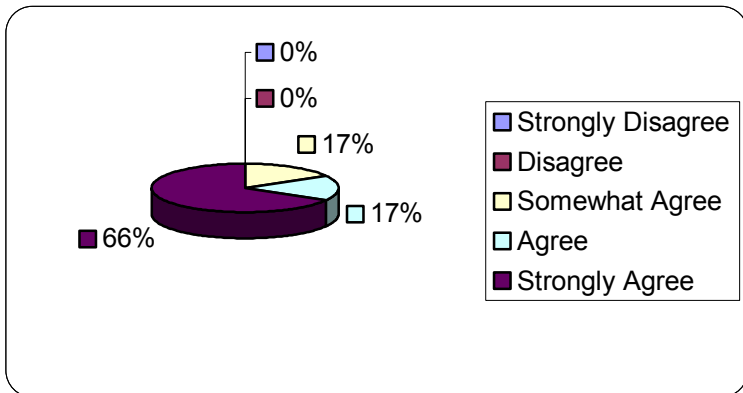
The program will generate all the reports requested by the client.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 1  
Agree: 1  
Strongly Agree: 4

**Agreement Rating**

4.5



**Comments**

“All reports may not make it into the app due to time constraints, but we will see.”

**Statement**

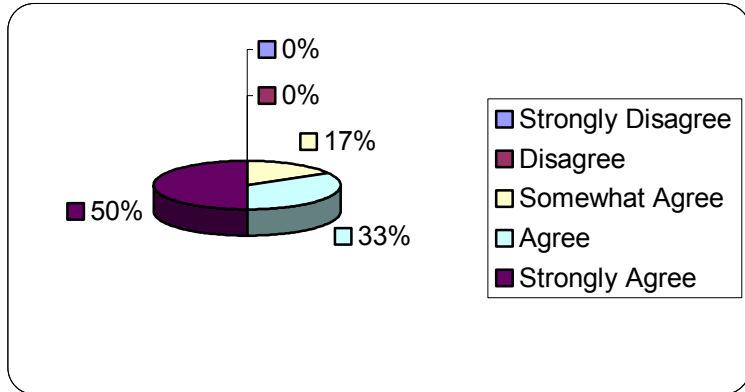
The Lookup screens are intuitive.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 1  
Agree: 2  
Strongly Agree: 3

**Agreement Rating**

4.3



**Statement**

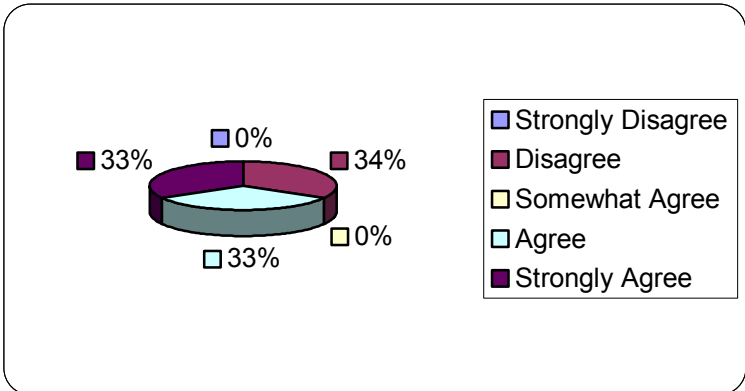
The lookup screens allow enough unique criteria to search by.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 2  
Somewhat Agree: 0  
Agree: 2  
Strongly Agree: 2

**Agreement Rating**

3.7



**Comments**

“The lookup screens currently do not allow enough criteria to search by. If we have time we will add that in for them all. Currently, only the invoice lookup has search criteria.”

**Statement**

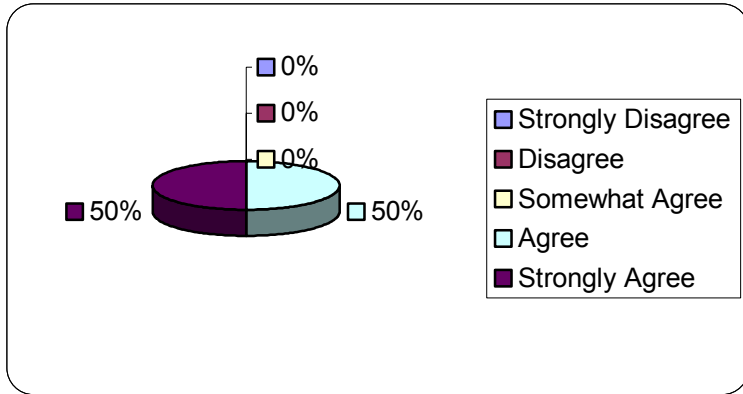
In the New Invoice screen, I want customer information to appear with the customer name.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 0  
Agree: 3  
Strongly Agree: 3

**Agreement Rating**

4.5



**Statement**

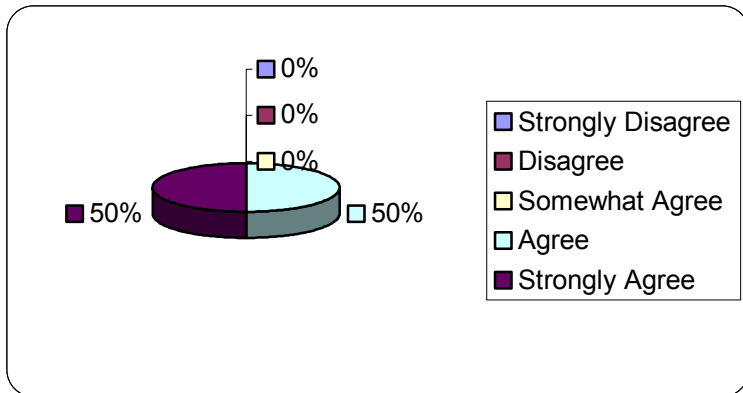
In the New Invoice screen, I want vehicle information to appear with the vehicle name.

**Number of Responses**

Strongly Disagree: 0  
Disagree: 0  
Somewhat Agree: 0  
Agree: 3  
Strongly Agree: 3

**Agreement Rating**

4.5



**Statement**

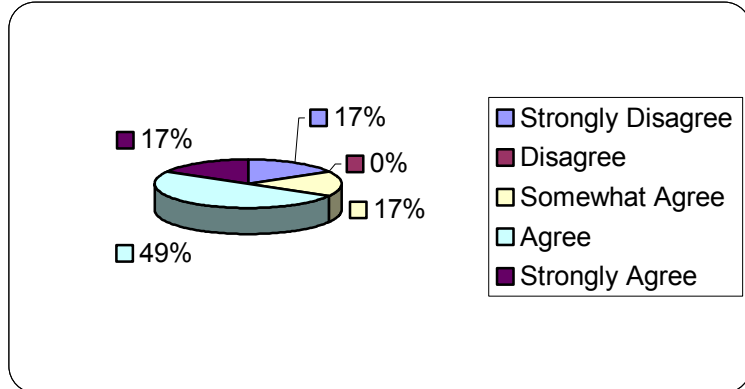
Adding line items in a separate window makes the New Invoice screen less confusing.

**Number of Responses**

Strongly Disagree: 1  
Disagree: 0  
Somewhat Agree: 1  
Agree: 3  
Strongly Agree: 1

**Agreement Rating**

3.5



**Comments**

“As an alternative to adding line items in a separate pop-up window, can line items be added directly on the invoice screen?”

“For now, we will keep the separate window for adding line items. I was not able to figure out the combo box inside of list view fields. If we have time, I will make that change (if team and/or sponsor agree).”

**Statement**

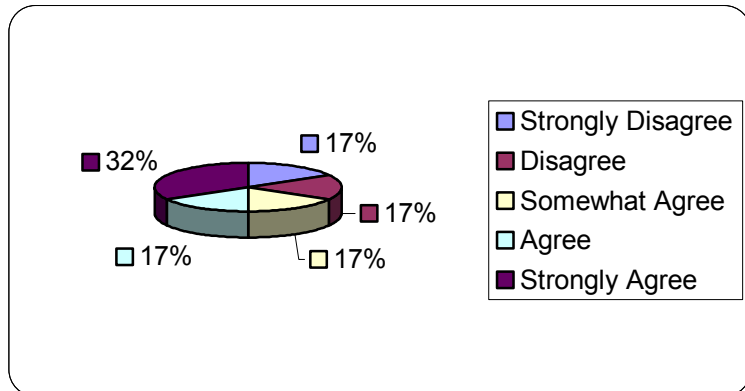
Adding line items in a separate window makes creating a new invoice easier.

**Number of Responses**

Strongly Disagree: 1  
Disagree: 1  
Somewhat Agree: 1  
Agree: 1  
Strongly Agree: 2

**Agreement Rating**

3.3



**Statement**

In the Lookup Invoice screen, I easily know what the letter buttons will do.

**Number of Responses**

Strongly Disagree: 0

Disagree: 3

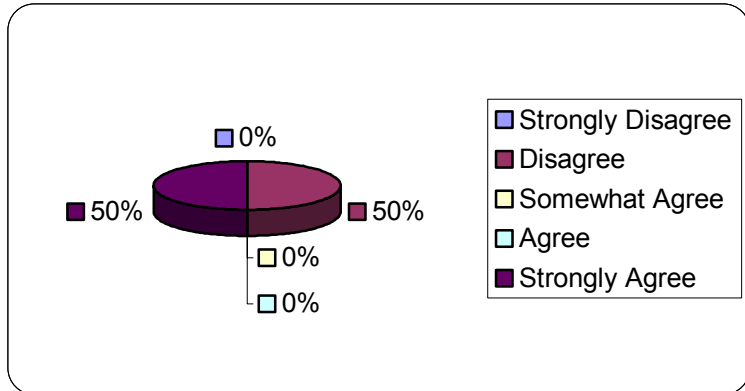
Somewhat Agree: 0

Agree: 0

Strongly Agree: 3

**Agreement Rating**

3.5



**Comments**

“Will these buttons sort on Customer Name? Vehicle Name”...its not very clear.”

”I took out the letter buttons on the invoice lookup. I don’t think they fit with that type of screen.”



## B.2 Sponsor Evaluation Results

Statement	Response				
	Strongly Disagree	Disagree	Somewhat Agree	Agree	Strongly Agree
<b>Section 1: Visual appearance</b>					
The colors used make the program easy to see and use.					
The fonts used make the program easy to see and use.					
Button and field labels are easy to read.					
Screens contain too much information; they are difficult to look at.					
I like to picture on the main screen.					
<b>Section 2: Program Layout</b>					
All of my most common tasks are easy to get to from the opening screen.					
All of the common tasks are easy to get to from most any screen I am in.					
The items in the top menu bar belong in the menu they are in.					
I can accurately predict what will happen for each button I click.					
It is easy to navigate through all the tasks I need to perform.					

	Strongly Disagree	Disagree	Somewhat Agree	Agree	Strongly Agree
<b>Section 3: Program Function</b>					
All of the tasks I need to perform are available.					
The Invoice screen contains all the information I want on an invoice.					
The program will generate all the reports I currently need (see Reports menu).					
The Lookup screens (for invoices, customers, etc.) allow me to search in ways I find useful.					

**Comments:**

“I’m amazed and impressed that after answering a couple of surveys a nearly finished product arrives that meets virtually all my invoicing and records needs.

“To know how to ask the right questions to get the right answers and then interpret that into an easy to use and comprehensive program is outstanding.”

## Appendix C

### Software Testing

A formal test case document was created for the testing phase of the STS project. This document contained a comprehensive set of procedures for the testers to follow. The results of these test cases are intended to drive the final modifications and fixes to the software. Below the results of the test cases can be seen. These results are very effective in demonstrating the functionality of the software, and stands as a definitive mark of this version of the software's quality. Comparison of these results to results from the same test cases for a future version of the software will prove effective for regression testing purposes. They will serve as a mark of what has or has not changed with the software.

#### C.1 Test Case Results

##### P00046 STS Software Test Cases

Filled out by: <u>Rima Patel and Tony Yang</u>	Date: <u>12/2/2002 (version beta F)</u>
Created by: <u>Richard Sonderfan</u>	Date: <u>11/26/2002</u>

### I. Introduction

This document contains the test cases that should be followed for complete testing of the STS application. Please read all the information before doing anything. The procedure to follow is: (1) read the test case and any necessary instructions, (2) record the date of your testing and also the current version number of the STS, (3) carry out the specified test with the software, (4) record any problems you see in the appropriate field. If not problems occurred, just type "OK" in the 'Results' field. You should not only be looking for actual error messages and other obvious errors, but also for weird behavior and things that could confuse the user. Please do not skip over any test cases simply

because the task is becoming too monotonous – the results from these tests will be very important.

## II. Known Issues

As this software is not yet complete, there are definitely going to be bugs in it. Here is a list of some of the known issues with the STS:

- There are NO reports in the software so far.
- None of the A-Z buttons on any of the lookup screens are implemented.
- On the ‘Invoice Lookup’ screen, the filter functionality has not yet been implemented.
- There may be some errors saving data to the database with a certain form if not ALL of the values are filled in, (e.g. leaving some fields blank and pressing ‘Save’ could result in errors).
- There currently is no checking for user mistakes on data entry.
- If a database record is manually entered via MS Access and certain fields are left blank (NULL), the STS will give errors when attempting to read that record.
- Tab-order is incorrect for EVERY screen.
- There can only be one invoice being created at a time – if you are in the middle of creating an invoice and press the ‘New’ Invoice button on the main screen all changes are lost.

## III. Test Cases

<b>ID</b>	<b>Description</b>	<b>Expected Outcome</b>	<b>Results (tester 1)</b>	<b>Results (tester 2)</b>
1	Main Screen – click every button and every menu option.	The correct screen should be opened for every button and menu option.	<ol style="list-style-type: none"> <li>1. The screen show up after click modify even when nothing has been selected.</li> <li>2. some couldn't detect wrong type of data entered and some cause the program to terminate</li> </ol>	OK
2	Add 5 new services. Have	The 5 services should get added	<ol style="list-style-type: none"> <li>1. Even when we didn't select</li> </ol>	Giving runtime type mismatch

	some of them use different warranties and some use a supplier and some use no supplier.	no problem. The list in the lookup screen is updated.	warranty or supplier, it still show a warranty item in the service lookup screen, but correct information in the modify screen	error Closed application
3	Add a new service, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	1. Error occurs when saving without entering miles or days	Only works if enter correct info other wise runtime error
4	Modify all of the existing services, (double click them in the lookup window and change some values, then click 'Save').	The services should be modified and the information in the lookup screen is updated after the save.	Same error as #2	ok
5	Delete 3 of the services.	The deleted services should be removed from the lookup screen list.	No error	ok
6	Add 5 new employees.	The 5 employees should get added no problem. The list in the lookup screen is updated.	No error	Ok if enter all the fields
7	Add a new employee, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	Error when not entering vacation days, tax withheld, and deduction	Giving runtime error
8	Modify all of the existing	The employees should be	Error when not entering vacation days, tax	ok

	employees, (double click them in the lookup window and change some values, then click 'Save').	modified and the information in the lookup screen is updated after the save.	withheld, and deduction	
9	Delete 3 of the employees.	The deleted employees should be removed from the lookup screen list.	No error	ok
10	Add 5 new employee histories to an existing employee.	The 5 employee histories should get added no problem. The list in the lookup screen is updated.	No error	Not adding
11	Add a new employee history record, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	When entering invalid data in salary and vacation days, the data becomes 0.	Not adding
12	Modify all of the existing employee histories for a single employee, (double click them in the lookup window and change some values, then click 'Save').	The employee histories should be modified and the information in the lookup screen is updated after the save.	No error, except change wont be made in the lookup window until we clicking on the employee name again.	Not modifying any information
13	Delete 3 of the employee histories for a single employee.	The deleted employee histories should be removed from the lookup screen list.	Delete not working.	Not deleting
14	Modify the Company Info.	New data should get saved	No error	ok

		properly.		
15	Modify Company Info, except leave some fields blank and/or supply invalid data.	The STS should give an error message and allow the user to correct problems.	Data becomes 0 when enter none-numerical data or no data in tax, hourly rate or invoice #.	It's letting you add and leave blank it's not giving any error
16	Add 5 new warranties.	The 5 warranties should get added no problem. The list in the lookup screen is updated.	No error	ok
17	Add a new warranty, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	Error when entering invalid data in days and miles.	ok
18	Modify all of the existing warranties, (double click them in the lookup window and change some values, then click 'Save').	The warranties should be modified and the information in the lookup screen is updated after the save.	Error when entering invalid data in days and miles.	ok
19	Delete 3 of the warranties.	The deleted warranties should be removed from the lookup screen list.	No error	ok
20	Add 5 new customers.	The 5 customers should get added no problem. The list in the lookup screen is updated.	No fax number field	ok
21	Add a new customer, but exclude certain information and enter some invalid info.	STS should prompt user with error and allow him/her to correct the mistakes.	No error. It doesn't prompt user to fill in all data.	ok
22	Modify all of the existing	The customers should be	When loading the modify screen, the	ok

	customers, (double click them in the lookup window and change some values, then click 'Save').	modified and the information in the lookup screen is updated after the save.	home phone number will be change to cell number.	
23	Delete 3 of the customers.	The deleted customers should be removed from the lookup screen list.	No error	ok
24	Add 5 new vehicles to an existing customer.	The 5 vehicles should get added no problem. The list in the lookup screen is updated.	No error	Runtime type mismatch error
25	Add a new vehicle, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	No error, except change wont be made in the look up window until we clicking on the customer name again.	Runtime error
26	Modify all of the existing vehicles for a single customer, (double click them in the lookup window and change some values, then click 'Save').	The vehicles should be modified and the information in the lookup screen is updated after the save.	No error, except change wont be made in the look up window until we clicking on the customer name again.	Not saving
27	Delete 3 of the vehicles for a single customer.	The deleted vehicles should be removed from the lookup screen list.	No error	I can't see any data
28	Add 5 new parts. Have some of them use different warranties and some use a supplier and some use no supplier.	The 5 parts should get added no problem. The list in the lookup screen is updated.	Missing part number field and the title of parts description is different in the new window.	Part number is missing in column header
29	Add a new part, but exclude	The STS should prompt user with	Data becomes 0 when entering invalid data or	ok



	certain information and enter some invalid info.	error and allow him/her to correct the mistakes.	blank in unit cost and price.	
30	Modify all of the existing parts, (double click them in the lookup window and change some values, then click 'Save').	The parts should be modified and the information in the lookup screen is updated after the save.	Data becomes 0 when entering invalid data or blank in unit cost and price.	Error invalid property index
31	Delete 3 of the parts.	The deleted parts should be removed from the lookup screen list.	No error	ok
32	Add 5 new suppliers.	The 5 suppliers should get added no problem. The list in the lookup screen is updated.	The order of the information is wrong on supplier look up page and shouldn't show supplier id.	ok
33	Add a new supplier, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	No error	ok
34	Modify all of the existing suppliers, (double click them in the lookup window and change some values, then click 'Save').	The suppliers should be modified and the information in the lookup screen is updated after the save.	No error	ok
35	Delete 3 of the suppliers.	The deleted suppliers should be removed from the lookup screen list.	Delete not working	Not deleting
36	Create 2 invoices. The remaining	(see all the remaining test	n/a	n/a

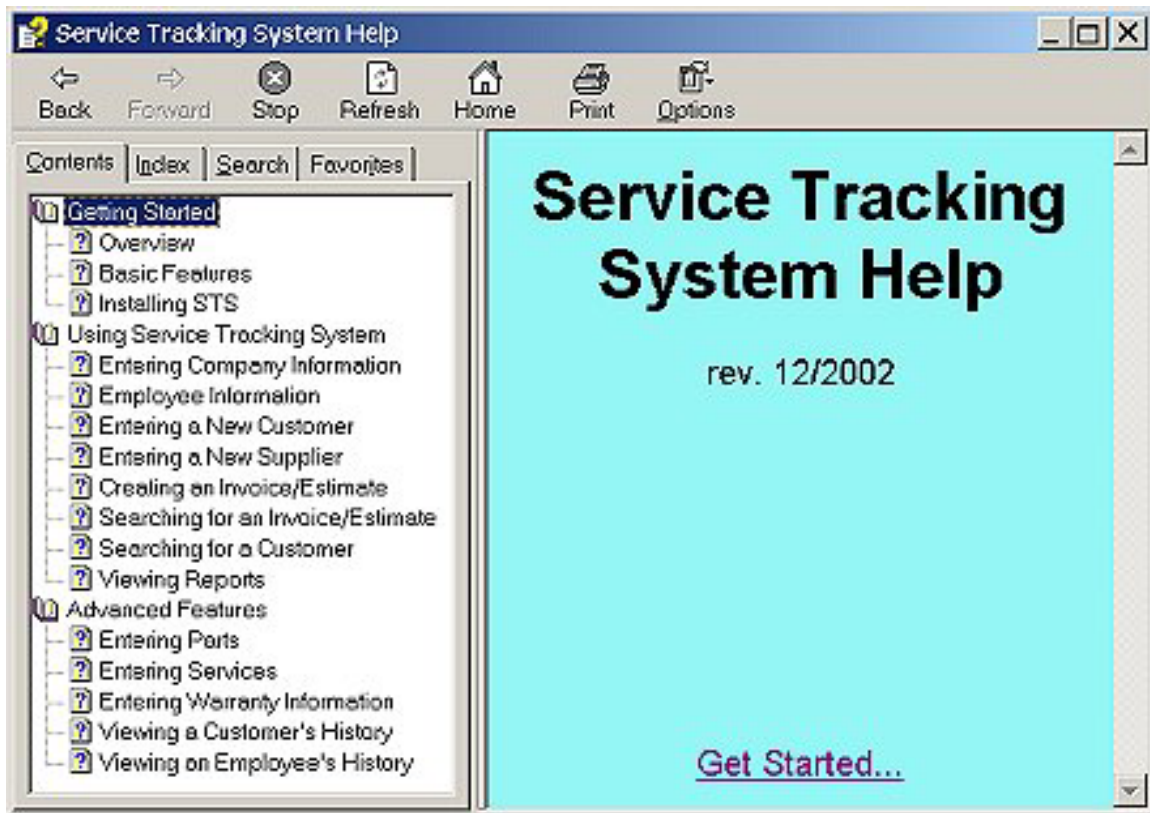
	test cases will cover this process more in-depth.	cases)		
37	For a new invoice, choose a customer from the dropdown.	The customer's info should be displayed below the combo, and all of his/her vehicles should be loaded to the vehicle combo	No error	ok
38	Choose a vehicle from the vehicles combo.	The vehicle's info should be displayed directly below the combo once it is selected.	No error. After selecting a customer, the new vehicle button didn't load a blank window with correct customer name.	ok
39	Enter the mileage in and out; choose an employee from the employee combo box.	These simple operations should be allowed.	Error saving new invoice when not add a line. Couldn't add a line.	ok
40	Add 5 new line items.	The 5 line items should get added no problem. The list in the invoice screen is updated. <b>AND</b> the money subtotals and total on the bottom right of the invoice should be adjusted after EACH line item is added.	new line in estimate screen cant be saved. the new part button didn't load a blank window. cant save a new part	ok
41	Add a new line item, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	Cant add a line item in new invoice but able to add a line in the modified invoice. After adding a line, the rest of information on the invoice is gone and need to be re-entered.	ok
42	Modify all of the existing line	The line items should be	Error when entering invalid data in	ok

	items, (double click them in the invoice window and change some values, then click 'Save').	modified and the information in the invoice screen is updated after the save.	numerical fields.	
43	Delete 3 of the line items.	The deleted line items should be removed from the invoice screen list.	No error	ok
44	Add 2 new payment records.	The 5 payment records should get added no problem. The list in the invoice screen is updated.	No error	ok
45	Add a new payment record, but exclude certain information and enter some invalid info.	The STS should prompt user with error and allow him/her to correct the mistakes.	Cant enter invalid data.	Runtime error
46	Modify all of the existing payment records, (double click them in the invoice window and change some values, then click 'Save').	The payment records should be modified and the information in the invoice screen is updated after the save.	Cant enter invalid data	ok
47	Delete 1 of the payment records.	The deleted payment record should be removed from the invoice list.	No error	ok
48	Modify an existing invoice, via the Invoice Lookup screen.	Should be able to modify successfully.	Error saving modified invoice sometimes.	ok
49	Delete an invoice via the Invoice Lookup screen.	Should be able to delete an invoice.	No error	ok

## Appendix D

### Help Documentation

The help documentation (help and user manual) is presented to the user as a Compiled HTML File, (CHM), and is supplied electronically on the installation CD. The STS software can launch the help file, or the user can double click it using Windows Explorer. Also, the user can print out the help file if he/she desires to have a hard copy. The HTML pages were compiled using BlueSky Software's RoboHelp HTML 2000 software package. The help file (shown below) has tabbed sections on the left, such as 'Contents', 'Index', 'Search', and 'Favorites'. These sections will help the user find exactly what he/she is looking for, in the quickest time possible.



# What is Service Tracking System?



## System Overview

STS is **Service Tracking System**, which allows an auto service provider to maintain a business using complete automated system. The most important feature of the system is its capability to generate invoice and bill a customer. In addition, the system allows management and storage of all customer and vehicle information, as well as services, parts, and supplier management. STS will also allow for complex financial [reports](#) and other quantitative reports.

## System Requirements

This section describes the hardware, system software, memory, and disk space requirements for using STS on a stand-alone computer or on a network.

### Hardware

STS requires the following hardware:

- Any Intel or compatible processor based computer with Microsoft ® Windows (any platform).
- 128 MB of memory.
- Minimum disk space requirement of 12.5 MB, with space to expand as the STS database grows.
- A VGA monitor.

# Major System Components

STS is designed around three primary components:

- Database (current application is utilizing Microsoft Access database), using ActiveX Data Objects.
- Windows graphical interface.

Click [here](#) to learn about some of the STS [basic features](#).

## Basic Features

STS is feature rich system and contains functionalities provided based on user specifications:

### Invoicing

Perhaps the most often used feature of this system allowing the user to generate [invoices](#) on demand while serving customers.

### Estimating

This is a very important feature allowing the user to give the customer a price [estimate](#) for a job quickly and efficiently.

### Reports

Several helpful [reports](#) are available for the user:

1. Parts sales per month and year
2. Labor sales tax per month and year
3. Parts sales tax per month and year
4. Total sales tax per month and year
5. Labor hours billed per employee per month and year
6. Sublets information (type and total price)
7. Number of each type of service performed per month and year
8. All invoices for a given month

### Maintenance

Daily maintenance of the system and data is available for the user:

1. Filling in your [Company Info](#)
2. Entering [Employee](#) Information

3. Entering a New [Customer](#)
4. Entering a New [Supplier](#)
5. Creating an [Invoice](#)
6. Creating an [Estimate](#)
7. Viewing a [Report](#)
8. [Searching](#) for an Invoice or Estimate
9. [Searching](#) for a Customer

## Entering Company Information

The screenshot shows a window titled "My Company Info" with the following fields and values:

- Company Name: Heights Service Station
- Address: 70 Glen Avenue
- City: Midland Park
- State: NJ
- Zipcode: 07432
- Phone Number: (201) 444-6753
- FAX Number: (201) 493-1444
- Website URL: (none)
- E-Mail Address: (none)
- Hourly Rate: 68.00
- Sales Tax: 0.06
- Invoice Start #: 5000
- Invoice Message 1: Thank you for your business. Have a great holiday season!
- Invoice Message 2: All parts and labor covered for 30 days or 2000 miles.

**In a newly installed system, Company Information stores the following information:**

- Company Name – name of business.
- Address, City, State, Zip
- Phone, Fax Number
- Website URL – company Website.
- Email Address – company email address.
- Hourly rate – **dollar amount the company charges for service (links to Estimate and Invoice).**
- Sales tax - sales tax percent sales tax.
- Invoice Start Number – **user can chose the identification number to start the first Invoice with. This is only for the initial start up. Once started subsequent invoices will be incremented accordingly. Invoice Startup Number is ideal for a business that wants to computerize their existing records.**
- Invoice Message 1 and Invoice Message 2 - **The information entered here will be linked to the Invoice screen and will be printed on the Invoice.**

## Employee Information

The screenshot shows a window titled "My Employees" with a table of employee data and an "Employee History" section below it.

First Name	Last Name	Address	City	State	Zip
Jack	Ripper	976 Evil Place	Scary	IL	09876
John	Smith	70 Mockingbird L...	Somewhe...	NJ	12345

Below the table is a horizontal scroll bar and an alphabetical index from A to Z. To the right of the index are buttons for "New", "Modify", and "Delete".

The "Employee History" section contains a table with columns for "Date" and "Comment", and buttons for "New", "Modify", and "Delete". A "Close" button is located at the bottom right of the window.

A, B, C, D, ..., Z- allows user to quickly scroll through the list of employees based on the first letter of employee's last name

New- allows user to add a new employee without closing the widow.

Modify- allows user to open the add/modify window without closing Employee Information window

Delete- allows user to delete employee information of the highlighted window

Close- Exits the window

Employee History-

New - user can add new information on employee

Modify - user can modify employee history information

Delete - user can delete employee history information for the highlighted employee in the above window

## Entering A New Customer

The screenshot shows a window titled "Add/Modify Customer Info" with a form for entering customer information.

The form includes the following fields:

- First Name
- Last Name
- Phone Number
- Address
- Fax Number
- City
- State (dropdown menu)
- Postal Code
- Cell Phone Number
- Email Address
- Work Number
- Comments (text area)

At the bottom of the form are "Save" and "Cancel" buttons.



First Name – **the first name of customer**  
 Last Name – **last name of customer**  
 Phone Number- **Customer’s phone number**  
 Fax Number- **Customer’s Fax Number**  
 Cell Phone Number- **Customer’s Cell phone number**  
 Work Number- **Customer’s Work Number**  
 City- **Customer’s City**  
 State- **User picks, from the drop down menu, the customer’s state**  
 Zip- **the zip code of**  
 Email Address- **Email Address of the customer**  
 Comments- **Comments on customer**  
 Save – **Saves information about customer in the database**  
 Cancel – **Cancels the information and exits the window**

## Entering A New Supplier

The image shows two overlapping windows from a software application. The 'Add/Modify Supplier Info' window on the left contains several input fields: 'Supplier Name', 'Address', 'City', 'State' (a dropdown menu), 'ZipCode', 'Phone Number', 'Fax Number', 'Email Address', 'Contact First Name', 'Contact Last Name', 'Contact Title', and a 'Comments' text area. At the bottom are 'Save' and 'Cancel' buttons. The 'Suppliers Lookup' window on the right displays a table of existing suppliers with columns for 'Supplier Name', 'Contact First Name', 'Contact Last Name', and 'Contact Title'. Below the table is an alphabetical index (A-Z) and buttons for 'New', 'Modify', 'Delete', and 'Close'.

Supplier Name	Contact First Name	Contact Last Name	Contact Title
King Auto Parts	John	Doe	Clerk
Midland Auto Parts	Don	Willy	Boos
SmoothWerks Body S...	Rico	Sauve	Body worker
Steve's Transmissions	Steve	Van Trasmission	Boos

Supplier Name- **Name of the business that supplies parts, (i.e. Manufacturer, Store, etc.)**  
 Address- **Address of the business**  
 Fax Number- **Fax number of the business**  
 Phone Number- **Phone number of business**  
 Email Address-**Email address of the business**  
 City- **City the business is located in**  
 State- **State the business is located in**  
 Zip Code- **Zip code of the business**  
 Contact First Name- **First name contact person**  
 Contact Last Name- **Last name contact person**  
 Contact Title- **Title of contact person**  
 Comments- **comments on the supplier**  
 Save- **saves the information on to the database**  
 Cancel- **cancels the information and exits the window**

## Creating An Invoice/Estimate

The screenshot shows a software window titled 'Invoice'. At the top, there are fields for 'Invoice #' (5009) and 'Invoice Date' (10/05/2002 09:54 PM). Below this are sections for 'Customer Info' and 'Vehicle Info'. The 'Customer Info' section shows 'Doe, John' with a 'New' button and address details. The 'Vehicle Info' section shows '2002 BMW 330ci' with a 'New' button and other vehicle details. To the right, there are 'Mileage In' (754) and 'Mileage Out' (767) fields, and an 'Employee' dropdown menu set to 'Jack Ripper'. The main part of the window is a 'Line Items' table with columns for Type, Description, Price Each, Quantity, and Total. It contains three rows: 'Sublet Body Work' (3 units, \$438.00 total), 'Labor Oil Change' (.5 units, \$34.00 total), and 'Parts Oil Filter' (1 unit, \$7.00 total). There are 'New', 'Edit', and 'Remove' buttons to the right of the table. Below the table is a 'Payments' section with a table for 'Date', 'Amount', and 'Method', and 'New', 'Edit', and 'Delete' buttons. On the right side, there is a summary of costs: 'Parts Total' (\$7.00), 'Labor Total' (\$472.00), 'Sub Total' (\$479.00), 'Tax (.06%)' (\$28.74), and 'Total' (\$507.74). At the bottom, there are 'Print Preview', 'Print', 'Save', and 'Close' buttons.

Invoice # - a number generated

Invoice Date- Updates the date automatically. A drop down menu displays the monthly calendar, allowing user to jump forward and back a month.

Customer Info- allows user to scroll through the customer list. Customer information is displayed for each customer.

New- allows user to add a new customer at the time of estimate without having to close/open additional windows.

Vehicle Info – scrolling through the vehicle list allows user to pick one of many vehicle a customer may have.

New- allows user to add a new vehicle at the time of estimate without having to close/open additional windows.

Mileage In – Mileage of Vehicle when it first arrives

Mileage Out – Mileage of Vehicle when it first arrives

Line Items –

New – creates a new Line Item

Edit – edits an existing Line Item

Remove – Deletes an existing Line Item

Payment-

New- Creates a new Payment History

Edit- Edits existing Payment History

Remove- Deletes the payment history

Parts Total- Gives total cost of parts used.

Labor Total- Gives total cost of labor used.

Subtotal- Calculates the total of labor cost and part cost.

Tax - **Linked with the Company Information where the state sales tax is filled out.**  
 Total- **Automated total with tax and Subtotal.**

## Searching For An Invoice Or Estimate

The screenshot shows a software window titled "Lookup Invoice/Estimate". It features a "Filters" section with several input fields and radio buttons. Below the filters is a table with columns for Invoice #, Date, Last Name, First Name, Type, Total, Amount P..., and Employee. At the bottom of the window are buttons for "New", "Modify", "Delete", and "Close".

Invoice #	Date	Last Name	First Name	Type	Total	Amount P...	Employee
5015	12/4/2002	Simpson	Homer	Invoice	\$608.18	\$0.00	Jack Pipp..
5014	12/3/2002	Flinstone	Fred	Invoice	\$75.77	\$0.00	John Smith
5013	12/1/2002	Simpson	Homer	Invoice	\$65.72	\$0.00	John Smith
5012	11/5/2002	Doe	Jane	Invoice	\$81.62	\$0.00	John Smith
5011	11/5/2002	Flinstone	Fred	Invoice	\$680.50	\$0.00	Jack Pipp..
5008	10/15/2002	Jackson	Michael	Invoice	\$531.06	\$0.00	John Smith
5010	10/5/2002	Simpson	Homer	Invoice	\$46.81	\$0.00	Jack Pipp..
5009	10/5/2002	Doe	John	Invoice	\$507.74	\$0.00	Jack Pipp..
5007	9/23/2002	Flinstone	Fred	Invoice	\$336.56	\$0.00	Jack Pipp..

### Filters

Last Name- **Last name of customer**

First Name- **First Name of customer**

Start Date- **Start date of the query. For example if user wants invoice and/or estimate information from January 1, 1995**

End Date- **End date of query. Allows user to pick the end date to find out invoice and/or estimate information. For example, if user would like to query invoice and/or estimate from a period beginning January 1, 1995 to December 31, 2000**

Service Performed- **allows user to query of specific service performed, such as oil change.**

Employee- **allows user to query based on a particular employee, allowing Manager to monitor employee's performance and/or projects**

Both- **Queries on Estimate and Invoice databases within a particular time, for a particular customer, with a particular employee, and for a particular service**

Invoice Only- **Queries databases within a particular time, for a particular customer, with a particular employee, and for a particular service for all the Invoices within the system.**

Estimates Only- **Queries databases within a particular time, for a particular customer, with a particular employee, and for a particular service for all the Invoices within the system.**

**New – Allows user to add an new invoice without closing the screen and opening a New Invoice Screen**

**Modify- Allows user to modify an existing invoice without closing this screen and opening modify Invoice screen**

**Delete- User can choose to delete an Invoice from within this screen.**

**Close- Allows the user to exit the screen.**

## Searching for a Customer

The screenshot shows a window titled "Customer/Vehicle Information". It contains two main sections: a customer list and a vehicle list. The customer list has columns for First Name, Last Name, Address, and City. The vehicle list has columns for Make, Model, Year, Color, and License. Both lists have "New", "Modify", and "Delete" buttons below them. A "Close" button is at the bottom right of the window.

First Name	Last Name	Address	City
Fred	Finstone	123 cartoon lane	RocksSomething
Homer	Simpson	123 Fake Street	Springfield
Jane	Doe	9376 Road Rd.	East Orange
John	Doe	947711 Street St.	Newark
Micheal	Jackson	492 ABC Street	Hollywood

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

New Modify Delete

Vehicles

Make	Model	Year	Color	Lic
Ford	Model-T	1754	Black	BL
Some	OtherCar	1000	Red	WH

New Modify Delete

Close

**A, B, C, D...- Allows user to scroll through the customer list by scrolling through the list by clicking on the first letter of the customer's last name.**

**Add- Opens the Add/Modify Customer Info information to add a new customer without closing the current window.**

**Modify- Open the Add/Modify Customer Info information to modify customer information without closing the current window.**

**Delete- Deletes the highlighted customer along with the vehicle information.**

**Vehicle Information**

**New- opens the window to Add/Modify Vehicle info to add a new vehicle**

**Modify- opens the window to Add/Modify Vehicle info to modify vehicle info.**

**Delete- deletes the highlighted vehicle for the particular customer.**

## Viewing A Report



Employee Labor Report Info

Start Date: 12/2002

End Date: 12/2002

Choose Employee: [Empty]

Go Cancel

Service Tracking System provides the ability to view detailed reports that are automatically generated from all of the information that is stored in your database. The following is a list of the available reports that you can use to better analyze your finances and other business specifics:

**Financial - A report the will show a generated list of all parts, labor, and sublet sales, parts, labor, and sublet sales tax, and total tax, total net and gross sales for each month.**

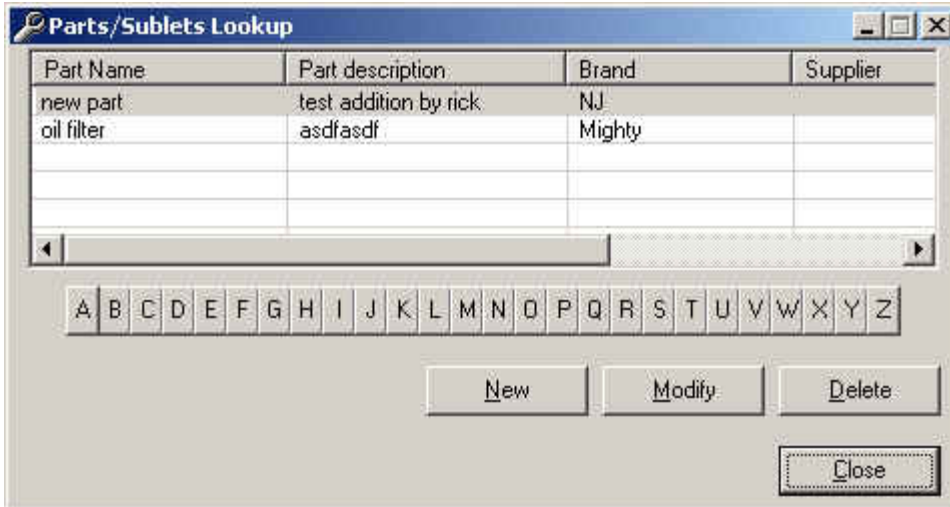
**Employee Labor - Generates a report, listing the amount of labor hours billed by each employee each month.**

**Services Performed - Generates a report on the amount of a given service or sublet performed each month.**

## Entering Parts

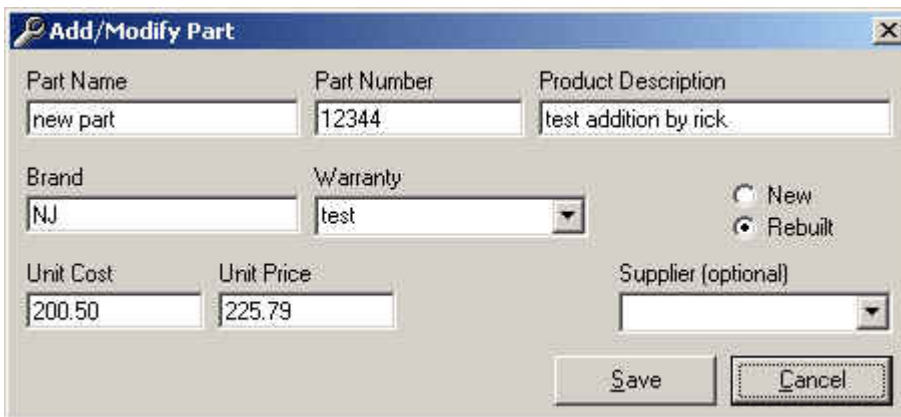
Maintaining an inventory of parts (does not necessarily pertain only to physical inventory but also what is required to perform a service) is one of the most fundamental tasks to maintain the auto service business. The pricing of a service may depend on the prices of parts.

To enter parts information go to the Resources section of the main application and choose parts. The following screen will appear:



### Modify existing Part

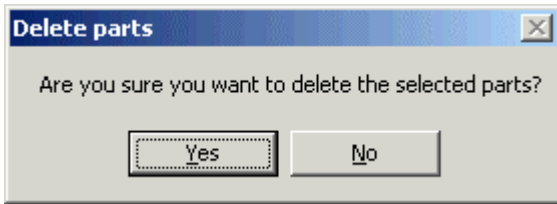
You may choose to view and modify an existing part by choosing from the index provided. Click on the part desired and then modify and the following screen will appear:



After modifying the data you may save or cancel.

### Delete existing Part

You may choose to delete an existing part by choosing from the index provided. Click on the part desired and then delete and the following screen will appear:



Click on Yes to delete or no to cancel.

## Add new Part

You may choose to add a new part by clicking on the NEW button and a blank screen will appear:

A form titled "Add/Modify Part" with a close button (X) in the top right corner. The form contains several input fields and controls: "Part Name", "Part Number", and "Product Description" (all text boxes); "Brand" (text box) and "Warranty" (dropdown menu); "Unit Cost" and "Unit Price" (text boxes); "Supplier (optional)" (dropdown menu); and two radio buttons labeled "New" (selected) and "Rebuilt". At the bottom right, there are "Save" and "Cancel" buttons.

After entering the information you may choose to save or cancel.

# Entering Services

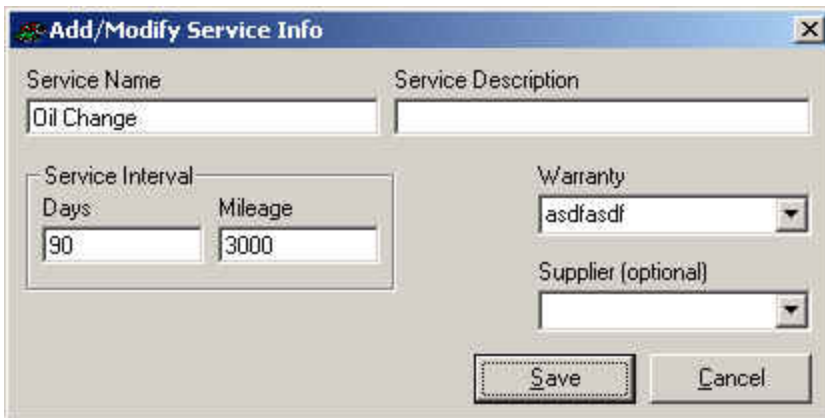
Maintaining an inventory of Services is one of the most fundamental tasks to maintain the auto service business.

To enter Service information, at the main application choose main of the menu on the top. From the drop down choose Services and the following screen will appear:



### Modify existing Service

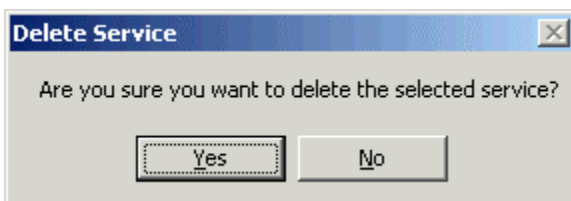
You may choose to view and modify an existing Service by choosing from the index provided. Click on the Service desired and then modify and the following screen will appear:



After modifying the data you may save or cancel.

### Delete existing Service

You may choose to delete an existing Service by choosing from the index provided. Click on the Service desired and then delete and the following screen will appear:





Click on Yes to delete or no to cancel.

## Add new Service

You may choose to add a new Service by clicking on the NEW button and a blank screen will appear:



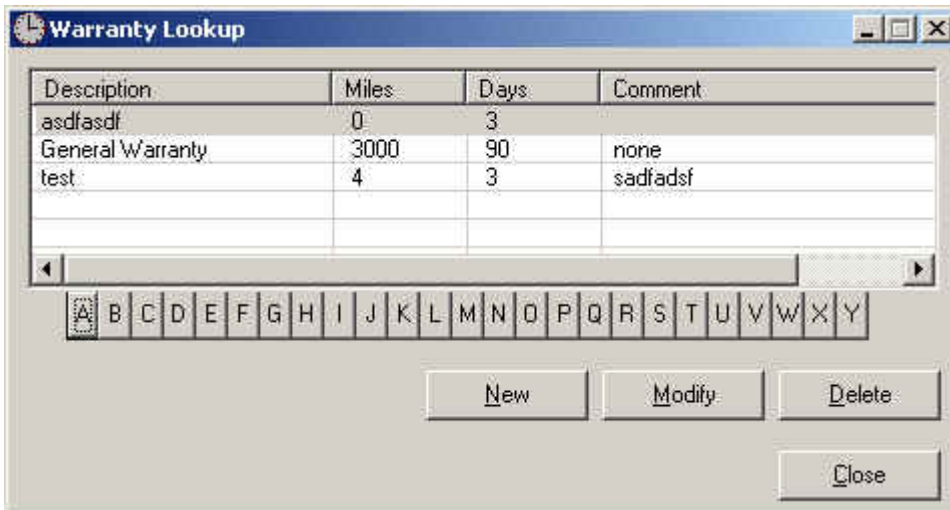
The screenshot shows a dialog box titled "Add/Modify Service Info". It has two main input fields at the top: "Service Name" and "Service Description". Below these, there is a "Service Interval" section with two sub-inputs: "Days" and "Mileage". To the right of the "Service Interval" section is a "Warranty" dropdown menu. Below the "Warranty" dropdown is a "Supplier (optional)" dropdown menu. At the bottom right of the dialog box are two buttons: "Save" and "Cancel".

After entering the information you may choose to save or cancel.

# Entering Warranty Information

Maintaining an inventory of Warranty is one of the most fundamental tasks to maintain the auto business.

To enter Warranty information from the main application and choose main of the menu on the top. From the drop down choose Warranty and the following screen will appear:



The screenshot shows a dialog box titled "Warranty Lookup". It contains a table with the following data:

Description	Miles	Days	Comment
asdfasdf	0	3	
General Warranty	3000	90	none
test	4	3	sadfadsf

Below the table is a scroll bar and an alphabetical index (A-Z). At the bottom of the dialog box are four buttons: "New", "Modify", "Delete", and "Close".

## Modify existing Warranty

You may choose to view and modify an existing Warranty by choosing from the index provided. Click on the Warranty desired and then modify and the following screen will appear:



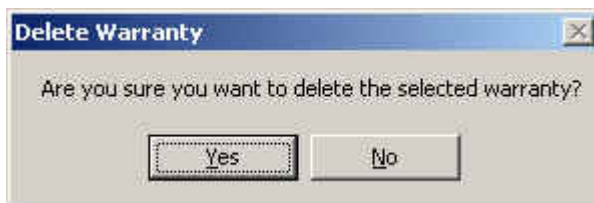
The screenshot shows a dialog box titled "Add/Modify Warranty Info". It contains the following fields and controls:

- Description:** A text box containing "General Warranty".
- Duration:** A section with two sub-fields: "Days" (containing "90") and "Mileage" (containing "3000").
- Comments:** A text box containing "none".
- Buttons:** "Save" and "Cancel" buttons at the bottom right.

After modifying the data you may save or cancel.

### Delete existing Warranty

You may choose to delete an existing Warranty by choosing from the index provided. Click on the Warranty desired and then delete and the following screen will appear:



The screenshot shows a dialog box titled "Delete Warranty". It contains the following elements:

- Text:** "Are you sure you want to delete the selected warranty?"
- Buttons:** "Yes" and "No" buttons at the bottom.

Click on Yes to delete or no to cancel.

### Add new Warranty

You may choose to add a new Warranty by clicking on the NEW button and a blank screen will appear:

**Add/Modify Warranty Info**

Description

Duration

Days      Mileage

Comments

Save      Cancel

After entering the information you may choose to save or cancel.

## Customer History

There may be occasion when the management may want to view the customer history to assess the customer profile. To view the customer use the Lookup button from the Customer section of the main application. Click the Lookup button and the following screen will appear:

**Customer/Vehicle Information**

First Name	Last Name	Address	City	State	Postal Co...	Phone
Duh	Sondafun333	(none)	asdfsdf	NJ	(none)	(none)
first	Last	(none)	somewhere	NJ	(none)	(none)
Rick	bla bla bla	(none)	a3erwer	NY	(none)	(none)
Rick	Sondafun	(none)	Hawthorne	NJ	(none)	(none)
Smith	Agent	123 Some St.	Matrix	MN	07654	555-1...

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z

New      Modify      Delete

**Vehicles**

Make	Model	Year	Color	Lic

New      Modify      Delete

Close

You may choose to view and modify an existing Customer by choosing from the index provided. This will provide the customer's vehicle profile. Next, you may see all the services the customer has purchased for

his vehicle(s) by looking up the invoices. You may choose the Lookup function from the Invoices section of the main application and the following screen will appear. Fill in the last name and/or first name and choose the desired customer.

**Lookup Invoice/Estimate**

Filters

Start Date: 11/03/2002 12:00 A  
 Customer Last Name:   
 Customer First Name:   
 End Date: 11/03/2002 12:00 A  
 Service Performed:   
 Employee:

Both  
 Invoices Only  
 Estimates Only

Clear Filters

Invoice #	Date	Last Name	First Name	Type	Total	Amount P...	Employee
5013	12/4/2002	bla bla bla	Rick	Invoice	\$497.14	\$0.00	Craig Son.
5012	12/4/2002	bla bla bla	Rick	Invoice	\$180.20	\$0.00	Craig Son.
5010	12/4/2002	bla bla bla	Rick	Invoice	\$46.64	\$0.00	Craig Son.
5009	12/4/2002	bla bla bla	Rick	Invoice	\$143.10	\$0.00	Craig Son.
5008	12/4/2002	Sondafun...	Duh	Invoice	\$156.88	\$0.00	Craig Son.
5007	12/4/2002	bla bla bla	Rick	Invoice	\$506.68	\$0.00	Craig Son.
5002	11/24/20...	Agent	Smith	Invoice	\$100.70	\$177.00	Craig Son.
5005	11/3/2002	Agent	Smith	Invoice	\$227.90	\$0.00	John Doe

New    Modify    Delete

Close

# Employee History

There may be occasion when the management may want to view the Employee history to assess the Employee profile. To view the Employee choose the main from the menu on the top and then choose Employees and the following screen will appear:

The screenshot shows a window titled "My Employees" with a table of employee data, an alphabetical index, and an "Employee History" section.

First Name	Last Name	Address	City	State	Zip
John	Doe	asdf	asdf	AZ	asdf
Craig	Sonderfan	asdf	asdf	nj	12345

Below the table is an alphabetical index from A to Z.

Buttons: New, Modify, Del

**Employee History**

Date	Comment

Buttons: New, Modify, Del, Close

You may choose to view and modify an existing Employee by choosing from the index provided. This will provide the Employee's profile. You may choose to modify an existing history and/or a new one or delete an existing history.

## Appendix E

### STS Source Code

The STS software was written using Microsoft Visual Basic 6.0 (VB), making use of ActiveX Data Objects (ADO) to connect to the database. A trial version of a third-party reporting package was used to create the reports, called ActiveReports 2.0. The following sections are each a separate file from the actual STS VB project. The version of the STS source code that is shown here is v1.01. Note that not all of the source code is shown here, only certain files which act as examples of the entire project.

#### E.1 Main Screen (frmMain.frm)



Option Explicit

```
Private Sub btnCustLookup_Click()  
    frmCustomers.Show ' show customers lookup dlg  
End Sub
```

```
Private Sub btnExit_Click()  
    Unload frmMain ' end the program when this form gets closed  
End Sub
```

```
Private Sub btnInvLookup_Click()  
    frmInvLookup.Show ' show the invoice form  
End Sub
```

```
Private Sub btnNewCustomer_Click()  
    frmNewCust.bNew = True  
    frmNewCust.Show 1 ' show add customer dlg
```

```

End Sub

Private Sub btnNewEstimate_Click()
    frmInvoice.bEstimate = True
    frmInvoice.bNew = True
    frmInvoice.Show ' show the invoice form
End Sub

Private Sub btnNewInvoice_Click()
    frmInvoice.bEstimate = False
    frmInvoice.bNew = True
    frmInvoice.Show ' show the invoice form
End Sub

Private Sub btnPartsLookup_Click()
    frmPartsLookup.Show ' show parts lookup dlg
End Sub

Private Sub btnSuppliersLookup_Click()
    frmSuppliersLookup.Show ' show parts lookup dlg
End Sub

Private Sub Form_Load()

    ' open the db on load
    StsOpenDatabase

End Sub

Private Sub Form_Unload(Cancel As Integer)
    StsCloseDatabase
    End ' end the program when this form gets closed
End Sub

Private Sub mnuAbout_Click()
    frmAbout.Show 1 ' show the About form (modal)
End Sub

Private Sub mnuCompanyInfo_Click()
    frmCompanyInfo.Show ' show this companies info
End Sub

Private Sub mnuContents_Click()
    ' open the help file
    If (Right(App.Path, 1) = "\") Then
        HtmlHelp Me.hWnd, App.Path + STS_HELP_FILENAME, HH_DISPLAY_TOPIC, 0
    Else
        HtmlHelp Me.hWnd, App.Path + "\" + STS_HELP_FILENAME, HH_DISPLAY_TOPIC, 0
    End If
End Sub

Private Sub mnuEmployeeLabor_Click()
    ' show the pre-report dialog, so user can enter filter criteria
    frmPreReport.nReportId = EMPLOYEE_LABOR_REPORT
    frmPreReport.Show 1
End Sub

```

```

Private Sub mnuEmployees_Click()
    frmEmployees.Show ' show list of employees
End Sub

Private Sub mnuExit_Click()
    End ' end program
End Sub

Private Sub mnuFinancial_Click()
    ' show the pre-report dialog, so user can enter filter criteria
    frmPreReport.nReportId = FINANCIAL_REPORT
    frmPreReport.Show 1
End Sub

Private Sub mnuNewCustomer_Click()
    frmNewCust.bNew = True
    frmNewCust.Show 1 ' show add customer dlg
End Sub

Private Sub mnuNewEmployee_Click()
    frmNewEmployee.bNew = True
    frmNewEmployee.Show 1 ' show dlg
End Sub

Private Sub mnuNewEstimate_Click()
    frmInvoice.bEstimate = True
    frmInvoice.bNew = True
    frmInvoice.Show ' show dlg
End Sub

Private Sub mnuNewInvoice_Click()
    frmInvoice.bEstimate = False
    frmInvoice.bNew = True
    frmInvoice.Show ' show dlg
End Sub

Private Sub mnuNewService_Click()
    frmNewService.bNew = True
    frmNewService.Show 1 ' show dlg
End Sub

Private Sub mnuNewSupplier_Click()
    frmNewSupplier.bNew = True
    frmNewSupplier.Show 1 ' show dlg
End Sub

Private Sub mnuServiceHistory_Click()
    ' show the pre-report dialog, so user can enter filter criteria
    frmPreReport.nReportId = SERVICES_REPORT
    frmPreReport.Show 1
End Sub

Private Sub mnuServices_Click()
    frmServices.Show ' show all services
End Sub

```

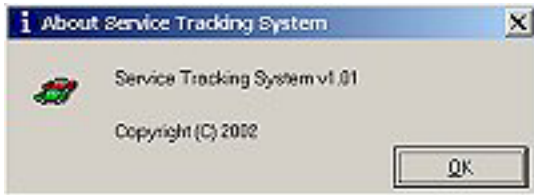


```

Private Sub mnuWarranty_Click()
    frmWarrantyLkup.Show ' show all warranties
End Sub

```

## E.2 About Screen (frmAbout.frm)



Option Explicit

```

Private Sub btnOK_Click()
    Me.Hide ' close this window when OK is clicked
End Sub

```

```

Private Sub Form_Load()

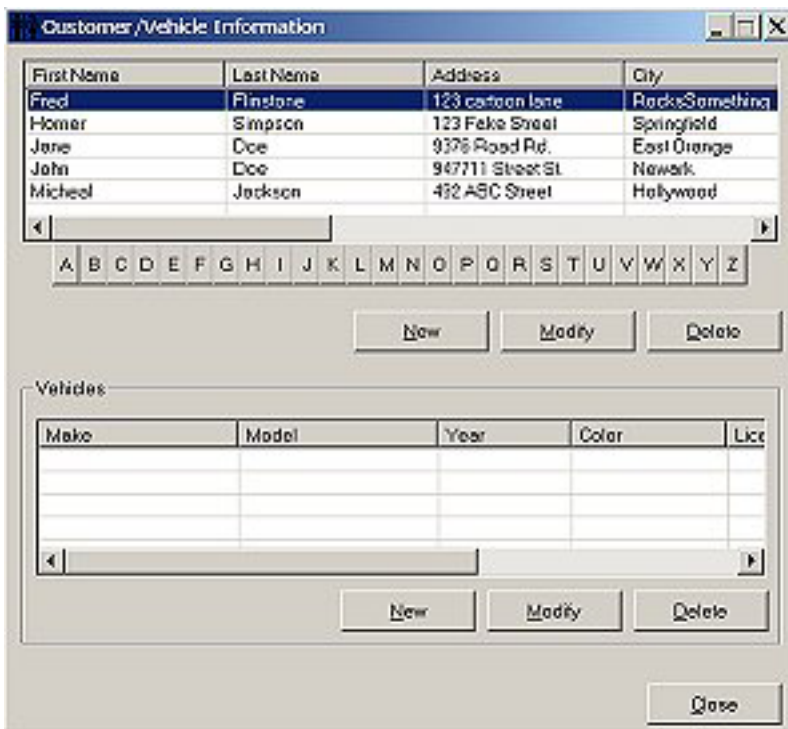
```

```

End Sub

```

## E.3 Customer Lookup Screen (frmCustomers.frm)



```

Option Explicit
Public nCurv As Integer
Public nCurSel As Integer

```

```

Private Sub btnClose_Click()
    Me.Hide ' close this window
End Sub

Private Sub btnDeleteCustomer_Click()
    ' prompt user before we actually do the delete
    If (MsgBox("Are you sure you want to delete the selected customer?", vbYesNo, "Delete
Customer") = vbYes) Then
        StsDeleteRecord CUSTOMER_TABLE, nCurSel ' delete current selection
        LoadList ' reload the list after we delete
    End If

End Sub

Private Sub btnDeleteVehicle_Click()
    ' prompt user before we actually do the delete
    If (MsgBox("Are you sure you want to delete the selected Vehicle?", vbYesNo, "Delete
Vehicle") = vbYes) Then
        StsDeleteRecord VEHICLE_TABLE, nCurv ' delete current selection
        Loadvehicle ' reload the list after we delete
    End If
End Sub

Private Sub btnModifyCustomer_Click()

    Dim sSQL As String

    sSQL = "SELECT * FROM CUSTOMER WHERE CUSTOMER_ID = " + Str$(nCurSel)
    If (StsGetRecords(sSQL, CUSTOMER_TABLE, False) = 1) Then
        frmNewCust.bNew = False
        frmNewCust.Show 1 ' show this dlg
    End If
End Sub

Private Sub btnModifyVehicle_Click()
    Dim sSQL As String

    sSQL = "SELECT * FROM VEHICLE WHERE VEHICLE_ID = " + Str$(nCurv)
    If (StsGetRecords(sSQL, VEHICLE_TABLE, False) = 1) Then
        frmNewVehicle.bNew = False
        frmNewVehicle.nCustID = nCurSel
        frmNewVehicle.Show 1 ' show this dlg
    End If
End Sub

Private Sub btnNewCustomer_Click()
    frmNewCust.bNew = True
    frmNewCust.Show 1 ' show dlg
End Sub

Private Sub btnNewVehicle_Click()
    frmNewVehicle.bNew = True
    frmNewVehicle.nCustID = nCurSel
    frmNewVehicle.Show 1 ' show dlg
End Sub

```

```

Private Sub Form_Activate()
    LoadList
End Sub
Private Function LoadList()
    Dim nCount As Integer
    Dim tCustomer As CUSTOMER

    ' first delete all combo items
    listCustomers.ListItems.Clear

    ' perform an SQL query to get all records, sorted by the main text field
    nCount = StsGetRecords("SELECT * FROM CUSTOMER ORDER BY FIRST_NAME ASC",
CUSTOMER_TABLE, True)

    ' loop through all records, loading info to listview
    While (StsReadCustomer(tCustomer, True))
        With listCustomers.ListItems.Add(, tCustomer.sFirstName)
            .Tag = tCustomer.nCustomerId ' set ID for this list row
            .ListSubItems.Add(, tCustomer.sLastName
            .ListSubItems.Add(, tCustomer.sAddress
            .ListSubItems.Add(, tCustomer.sCity
            .ListSubItems.Add(, tCustomer.sState
            .ListSubItems.Add(, tCustomer.sPostalCode
            .ListSubItems.Add(, tCustomer.sPhoneNumber
            .ListSubItems.Add(, tCustomer.sWorkPhoneNo
            .ListSubItems.Add(, tCustomer.sCellPhoneNumber
            .ListSubItems.Add(, tCustomer.sEmailAddress

            If (Len(tCustomer.sComments) <> 0) Then
                .ListSubItems.Add(, tCustomer.sComments
            End If
        End With
    Wend ' end while loop
End Function

Private Sub Form_Load()
    nCurSel = 0
    listCustomers.ColumnHeaders.Add(, "First Name", 1800).Tag = "FName"
    listCustomers.ColumnHeaders.Add(, "Last Name", 1800).Tag = "LName"
    listCustomers.ColumnHeaders.Add(, "Address", 1800).Tag = "ADDRESS"
    listCustomers.ColumnHeaders.Add(, "City", 1800).Tag = "CITY"
    listCustomers.ColumnHeaders.Add(, "State", 800).Tag = "STATE"
    listCustomers.ColumnHeaders.Add(, "Postal Code", 1000).Tag = "POSTCODE"
    listCustomers.ColumnHeaders.Add(, "Phone Number", 1500).Tag = "PHONE"
    listCustomers.ColumnHeaders.Add(, "Work Number", 1500).Tag = "WORK"
    listCustomers.ColumnHeaders.Add(, "Cell Phone Number", 1500).Tag = "CELL"
    listCustomers.ColumnHeaders.Add(, "Email Address", 1500).Tag = "EMAIL"
    listCustomers.ColumnHeaders.Add(, "Comments", 2000).Tag = "COMM"
    ,

    listVehicles.ColumnHeaders.Add(, "Make", 1800).Tag = "MAKE"
    listVehicles.ColumnHeaders.Add(, "Model", 1800).Tag = "MODEL"
    listVehicles.ColumnHeaders.Add(, "Year", 1200).Tag = "YEAR"
    listVehicles.ColumnHeaders.Add(, "Color", 1400).Tag = "COLOR"
    listVehicles.ColumnHeaders.Add(, "License Plate", 1500).Tag = "PLATE"
    listVehicles.ColumnHeaders.Add(, "VIN", 1500).Tag = "VIN"
    listVehicles.ColumnHeaders.Add(, "Engine Size", 1500).Tag = "ENGINE"

```

End Sub

Private Function Loadvehicle()

Dim nCount As Integer

Dim tVehicle As VEHICLE

Dim sSQLv As String

sSQLv = "SELECT \* FROM VEHICLE WHERE CUSTOMER\_ID = " + Str\$(nCurSel)

' first delete all combo items

listVehicles.ListItems.Clear

' perform an SQL query to get all records, sorted by the main text field

nCount = StsGetRecords(sSQLv, VEHICLE\_TABLE, True)

' loop through all records, loading info to listview

While (StsReadVehicle(tVehicle, True))

With listVehicles.ListItems.Add(, , tVehicle.sVehicleMake)

.Tag = tVehicle.nVehicleId

.ListSubItems.Add(, , tVehicle.sVehicleModel

.ListSubItems.Add(, , Str\$(tVehicle.nVehicleYear)

.ListSubItems.Add(, , tVehicle.sVehicleColor

.ListSubItems.Add(, , tVehicle.sLicensePlate

.ListSubItems.Add(, , tVehicle.sVin

.ListSubItems.Add(, , Str\$(tVehicle.fEngineSize)

End With

Wend ' end while loop

End Function

Private Sub listCustomers\_DbClick()

btnModifyCustomer\_Click

End Sub

Private Sub listCustomers\_ItemClick(ByVal Item As MSComctlLib.ListItem)

nCurSel = Item.Tag

Loadvehicle

End Sub

Private Sub listVehicles\_DbClick()

btnModifyVehicle\_Click

End Sub

Private Sub listVehicles\_ItemClick(ByVal Item As MSComctlLib.ListItem)

nCurv = Item.Tag

End Sub

## E.4 New Customer Screen (frmNewCust.frm)

The screenshot shows a Windows-style dialog box titled "Add/Modify Customer Info". It contains the following fields and controls:

- First Name: text box
- Last Name: text box
- Phone Number: text box
- Address: text box
- Fax Number: text box
- City: text box
- State: dropdown menu
- Postal Code: text box
- Cell Phone Number: text box
- Email Address: text box
- Work Number: text box
- Comments: text area with scrollbars
- Save: button
- Cancel: button

Option Explicit

Public bNew As Boolean

Public nCustomerId As Integer

Private Sub btnCancel\_Click()

Me.Hide ' close this window

End Sub

Private Sub btnSave\_Click()

Dim tCustomer As CUSTOMER

tCustomer.sFirstName = txtFName

tCustomer.sLastName = txtLName

tCustomer.sAddress = txtAddress

tCustomer.sCity = txtCity

tCustomer.sState = cobState

tCustomer.sPostalCOde = txtPostalcode

tCustomer.sEmailAddress = txtEmail

tCustomer.sComments = txtComments

tCustomer.sPhoneNumber = txtPhone

tCustomer.sCellPhoneNumber = txtCell

tCustomer.sFaxNumber = txtFax

tCustomer.sWorkPhoneNo = txtWork

tCustomer.nCustomerId = nCustomerId

If (StsWriteCustomer(tCustomer, bNew) = False) Then

MsgBox "Error Writing to the Database.", , "Add/Modify Customer Info"

Else

Me.Hide

End If

End Sub

Private Sub Form\_Activate()

Dim tCustomer As CUSTOMER

If (bNew = True) Then

nCustomerId = -1

```

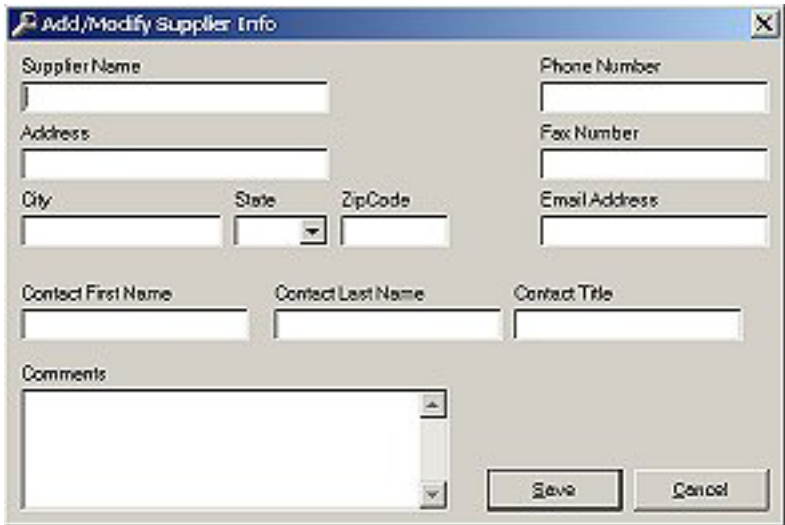
txtFName = ""
txtLName = ""
txtAddress = ""
txtCity = ""
cobState = ""
txtPostalcode = ""
txtEmail = ""
txtComments = ""
txtPhone = ""
txtCell = ""
txtFax = ""
txtWork = ""
Else
  If (StsReadCustomer(tCustomer, False)) Then

    txtFName = tCustomer.sFirstName
    txtLName = tCustomer.sLastName
    txtAddress = tCustomer.sAddress
    txtCity = tCustomer.sCity
    cobState = tCustomer.sState
    txtPostalcode = tCustomer.sPostalCOde
    txtPhone = tCustomer.sPhoneNumber
    txtFax = tCustomer.sFaxNumber
    txtPhone = tCustomer.sCellPhoneNumber
    txtWork = tCustomer.sWorkPhoneNo
    txtEmail = tCustomer.sEmailAddress
    txtComments = tCustomer.sComments
    nCustomerId = tCustomer.nCustomerId
  End If
End If

End Sub

```

## E.5 New Vehicle Screen (frmNewVehicle.frm)



Option Explicit  
Public nCustID As Integer

```

Public bNew As Boolean
Public nVehicleId As Integer

Private Sub btnCancel_Click()
    Me.Hide ' close this window
End Sub

Private Sub btnSave_Click()
Dim tVehicle As VEHICLE

    tVehicle.sVehicleMake = txtMake
    tVehicle.sVehicleModel = txtModel
    If (txtYear = "") Then
        tVehicle.nVehicleYear = 0
    Else
        tVehicle.nVehicleYear = txtYear
    End If
    tVehicle.sVehicleColor = txtColor
    tVehicle.sLicensePlate = txtLicense
    tVehicle.sVin = txtVIN
    If (txtEngine = "") Then
        tVehicle.fEngineSize = 0
    Else
        tVehicle.fEngineSize = txtEngine
    End If
    tVehicle.nCustomerId = nCustID
    If (StsWriteVehicle(tVehicle, bNew) = False) Then
        MsgBox "Error Writing to the Database.", , "Add/Modify Vehicle Info"
    Else
        Me.Hide
    End If

End Sub

Private Sub Form_Activate()
Dim sSQL As String
Dim nCount As Integer
Dim tCustomer As CUSTOMER
sSQL = "SELECT * FROM CUSTOMER WHERE CUSTOMER_ID = " + Str$(nCustID)
nCount = StsGetRecords(sSQL, CUSTOMER_TABLE, True)
If (StsReadCustomer(tCustomer, False)) Then
    txtCust = tCustomer.sFirstName + " " + tCustomer.sLastName
End If

Dim tVehicle As VEHICLE

' load controls differently based on if we are adding new
' or showing an existing entry
If (bNew = True) Then
    nVehicleId = -1
    ' clear out controls for new
    txtMake = ""
    txtModel = ""
    txtYear = ""
    txtColor = ""
    txtEngine = ""

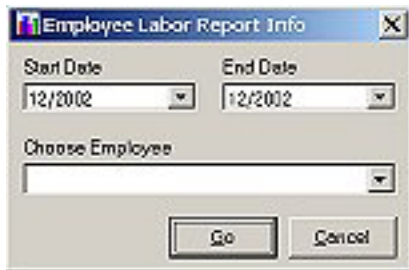
```

```

txtLicense = ""
txtVIN = ""
Else
' load current record into form
If (StsReadVehicle(tVehicle, False)) Then
txtMake = tVehicle.sVehicleMake
txtModel = tVehicle.sVehicleModel
txtYear = tVehicle.nVehicleYear
txtColor = tVehicle.sVehicleColor
txtEngine = tVehicle.fEngineSize
txtLicense = tVehicle.sLicensePlate
txtVIN = tVehicle.sVin
nVehicleId = tVehicle.nVehicleId
End If
End If
End Sub

```

## E.6 Pre-Report Screen (frmPreReport.frm)



```

Option Explicit
Public nReportId As Integer

```

```

Private Sub btnCancel_Click()
Me.Hide ' hide this window when cancelled
End Sub

```

```

Private Sub btnGo_Click()
Dim dStart As Date
Dim dEnd As Date
Dim sSQL As String
Dim nEmployeeId As Integer
Dim nServiceId As Integer

```

```

' execute the report
Select Case (nReportId)
Case FINANCIAL_REPORT:
dStart = dtpStart
dEnd = dtpEnd
' NOW CONNECT THE REPORT TO THE DATABASE AND SHOW THE REPORT
,

rptFinance.dtaDB.ConnectionString = gAdoConnectStr
sSQL = "SELECT DISTINCT dd.CUR_MONTH, dd.CUR_YEAR FROM [SELECT
MONTH(dd.INVOICE_DATE) as CUR_MONTH, YEAR(dd2.INVOICE_DATE) as CUR_YEAR
FROM INVOICE dd2 " + _
"WHERE (MONTH(dd2.INVOICE_DATE) >= " + Str(Month(dStart)) + _
") AND (YEAR(dd2.INVOICE_DATE) >= " + Str(Year(dStart)) + _

```



```

    ") AND (MONTH(dd2.INVOICE_DATE) <= " + Str(Month(dEnd)) + _
    ") AND (YEAR(dd2.INVOICE_DATE) <= " + Str(Year(dEnd)) + _
    ")] AS dd ORDER BY dd.CUR_YEAR, dd.CUR_MONTH"
rptFinance.dtaDB.Source = sSQL
rptFinance.dtaDB.Refresh
rptFinance.Restart
rptFinance.InitLabels dStart, dEnd
Me.Hide ' hide the pre-report dialog first
rptFinance.Show ' show the report
Case EMPLOYEE_LABOR_REPORT:
    dStart = dtpStart
    dEnd = dtpEnd
    If (cboCombo.ListIndex = -1) Then
        MsgBox "No employee was selected. You must first select an employee.", vbOKOnly,
"Must Select Employee"
        cboCombo.SetFocus
        Exit Sub
    Else
        nEmployeeId = cboCombo.ItemData(cboCombo.ListIndex)
    End If
    ' NOW CONNECT THE REPORT TO THE DATABASE AND SHOW THE REPORT
    ,
    rptEmployeeLabor.dtaDB.ConnectionString = gAdoConnectStr
    sSQL = "SELECT DISTINCT dd.CUR_MONTH, dd.CUR_YEAR FROM [SELECT
MONTH(dd2.INVOICE_DATE) as CUR_MONTH, YEAR(dd2.INVOICE_DATE) as CUR_YEAR
FROM INVOICE dd2 " + _
    "WHERE (MONTH(dd2.INVOICE_DATE) >= " + Str(Month(dStart)) + _
    ") AND (YEAR(dd2.INVOICE_DATE) >= " + Str(Year(dStart)) + _
    ") AND (MONTH(dd2.INVOICE_DATE) <= " + Str(Month(dEnd)) + _
    ") AND (YEAR(dd2.INVOICE_DATE) <= " + Str(Year(dEnd)) + _
    ")] AS dd ORDER BY dd.CUR_YEAR, dd.CUR_MONTH"
    rptEmployeeLabor.dtaDB.Source = sSQL
    rptEmployeeLabor.dtaDB.Refresh
    rptEmployeeLabor.Restart
    rptEmployeeLabor.InitLabels nEmployeeId, dStart, dEnd
    Me.Hide ' hide the pre-report dialog first
    rptEmployeeLabor.Show ' show the report
    '
    Case SUBLET_REPORT:
    Case SERVICES_REPORT:
        dStart = dtpStart
        dEnd = dtpEnd
        If (cboCombo.ListIndex = -1) Then
            MsgBox "No service was selected. You must first select a service.", vbOKOnly, "Must
Select Service"
            cboCombo.SetFocus
            Exit Sub
        Else
            nServiceId = cboCombo.ItemData(cboCombo.ListIndex)
        End If
        ' NOW CONNECT THE REPORT TO THE DATABASE AND SHOW THE REPORT
        ,
        rptServices.dtaDB.ConnectionString = gAdoConnectStr
        sSQL = "SELECT DISTINCT dd.CUR_MONTH, dd.CUR_YEAR FROM [SELECT
MONTH(dd2.INVOICE_DATE) as CUR_MONTH, YEAR(dd2.INVOICE_DATE) as CUR_YEAR
FROM INVOICE dd2 " + _
        "WHERE (MONTH(dd2.INVOICE_DATE) >= " + Str(Month(dStart)) + _

```

```

        ") AND (YEAR(dd2.INVOICE_DATE) >= " + Str(Year(dStart)) + _
        ") AND (MONTH(dd2.INVOICE_DATE) <= " + Str(Month(dEnd)) + _
        ") AND (YEAR(dd2.INVOICE_DATE) <= " + Str(Year(dEnd)) + _
        ")]. AS dd ORDER BY dd.CUR_YEAR, dd.CUR_MONTH"
rptServices.dtaDB.Source = sSQL
rptServices.dtaDB.Refresh
rptServices.Restart
rptServices.InitLabels nServiceId, dStart, dEnd
Me.Hide ' hide the pre-report dialog first
rptServices.Show ' show the report
End Select
End Sub

Private Sub Form_Activate()
    lblStart.Visible = False
    lblEnd.Visible = False
    dtpStart.Visible = False
    dtpEnd.Visible = False
    lblCombo.Visible = False
    cboCombo.Visible = False

    ' setup this window based on report type
    Select Case (nReportId)
        Case FINANCIAL_REPORT:
            lblStart.Visible = True
            lblEnd.Visible = True
            dtpStart.Visible = True
            dtpEnd.Visible = True
            Me.Caption = "Financial Report Info"
        Case EMPLOYEE_LABOR_REPORT:
            lblStart.Visible = True
            lblEnd.Visible = True
            dtpStart.Visible = True
            dtpEnd.Visible = True
            lblCombo.Visible = True
            cboCombo.Visible = True
            lblCombo = "Choose Employee"
            LoadEmployees ' load combo box
            Me.Caption = "Employee Labor Report Info"
        Case SUBLET_REPORT:
            Me.Caption = "Sublet Report Info"
        Case SERVICES_REPORT:
            lblStart.Visible = True
            lblEnd.Visible = True
            dtpStart.Visible = True
            dtpEnd.Visible = True
            lblCombo.Visible = True
            cboCombo.Visible = True
            lblCombo = "Choose Service/Sublet"
            LoadServices ' load combo box
            Me.Caption = "Services Report Info"
        Case Else:
            MsgBox "Unknown report.", vbOKOnly, "Pre-Report Window"
            Me.Hide
    End Select
End Sub

```

```

Private Sub Form_Load()
    ' assign current date to the date fields only on startup
    dtpStart = Now
    dtpEnd = Now
End Sub

' loads employee names to the combo
Private Function LoadEmployees()
    Dim nCount As Integer
    Dim tEmployee As EMPLOYEE

    ' first delete all list items
    cboCombo.Clear

    ' perform an SQL query to get all records, sorted by the main text field
    nCount = StsGetRecords("SELECT * FROM EMPLOYEE ORDER BY LAST_NAME ASC",
EMPLOYEE_TABLE, True)

    ' loop through all records, loading info to listview
    While (StsReadEmployee(tEmployee, True))
        cboCombo.AddItem (tEmployee.sFirstName + " " + tEmployee.sLastName)
        cboCombo.ItemData(cboCombo.NewIndex) = tEmployee.nEmployeeId
    Wend ' end while loop
End Function

' loads service names to the combo
Private Function LoadServices()
    Dim nCount As Integer
    Dim tService As SERVICE

    ' first delete all list items
    cboCombo.Clear

    ' perform an SQL query to get all records, sorted by the main text field
    nCount = StsGetRecords("SELECT * FROM SERVICE ORDER BY SERVICE_NAME ASC",
SERVICE_TABLE, True)

    ' loop through all records, loading info to listview
    While (StsReadService(tService, True))
        cboCombo.AddItem (tService.sServiceName)
        cboCombo.ItemData(cboCombo.NewIndex) = tService.nServiceId
    Wend ' end while loop
End Function

```

## E.7 STS Database Utilities (DatabaseUtils.bas)

This section only contains the top portion of this file, to show what the main set of database functions implemented are. This whole file, including this top portion and all of the implementation was over 2000 lines of code.

```
' DatabaseUtils.BAS - updated 11/14/2002
,
' This file is a global module that will provide the
' entire project with the data objects and utilities for
' interacting with the STS database.
Option Explicit

*****Begin*****
' The list of STS database utility functions that we will use:
,
' StsOpenDatabase () as Boolean
' StsCloseDatabase ()
' StsGetRecords (sSQL as String, nTableId as Integer, bCount As Boolean) as Integer
' StsRefreshRecords (nTableId as Integer) as Boolean
' StsMoveFirst (nTableId as Integer) as Boolean
' StsMoveLast (nTableId as Integer) as Boolean
' StsMoveNext (nTableId as Integer) as Boolean
' StsMovePrev (nTableId as Integer) as Boolean
' StsDeleteRecord (nTableId as Integer, nPrimaryKeyId As Integer) as Boolean
,
' ** Set of read functions (one per table):
' StsReadCustomer(tCustomer As CUSTOMER, bReadAndMove As Boolean) As Boolean
' StsReadVehicle(tVehicle As VEHICLE, bReadAndMove As Boolean) As Boolean
' StsReadSupplier(tSupplier As SUPPLIER, bReadAndMove As Boolean) As Boolean
' StsReadParts(tParts As PARTS, bReadAndMove As Boolean) As Boolean
' StsReadService(tService As SERVICE, bReadAndMove As Boolean) As Boolean
' StsReadEmployee(tEmployee As EMPLOYEE, bReadAndMove As Boolean) As Boolean
' StsReadEmpProfile(tEmpProfile As EMPLOYEE_PROFILE, bReadAndMove As Boolean) As
Boolean
' StsReadCoProfile(tCoProfile As COMPANY_PROFILE, bReadAndMove As Boolean) As
Boolean
' StsReadInvoice(tInvoice As INVOICE, bReadAndMove As Boolean) As Boolean
' StsReadInvDetails(tInvDetails As INVOICE_DETAILS, bReadAndMove As Boolean) As Boolean
' StsReadPayHistory(tPayHistory As PAYMENT_HISTORY, bReadAndMove As Boolean) As
Boolean
' StsReadWarranty(tWarranty As WARRANTY, bReadAndMove As Boolean) As Boolean
,
' ** Set of write functions (one per table):
' StsWriteCustomer(tCustomer As CUSTOMER, bNew As Boolean) As Boolean
' StsWriteVehicle(tVehicle As VEHICLE, bNew As Boolean) As Boolean
' StsWriteSupplier(tSupplier As SUPPLIER, bNew As Boolean) As Boolean
' StsWriteParts(tParts As PARTS, bNew As Boolean) As Boolean
' StsWriteService(tService As SERVICE, bNew As Boolean) As Boolean
' StsWriteEmployee(tEmployee As EMPLOYEE, bNew As Boolean) As Boolean
' StsWriteEmpProfile(tEmpProfile As EMPLOYEE_PROFILE, bNew As Boolean) As Boolean
' StsWriteCoProfile(tCoProfile As COMPANY_PROFILE, bNew As Boolean) As Boolean
' StsWriteInvoice(tInvoice As INVOICE, bNew As Boolean) As Boolean
' StsWriteInvDetails(tInvDetails As INVOICE_DETAILS, bNew As Boolean) As Boolean
' StsWritePayHistory(tPayHistory As PAYMENT_HISTORY, bNew As Boolean) As Boolean
```

```
' StsWriteWarranty(tWarranty As WARRANTY, bNew As Boolean) As Boolean
,
*****End*****
```

```
' Constants used to determine which DB table to communicate to
Public Const CUSTOMER_TABLE = 1
Public Const VEHICLE_TABLE = 2
Public Const SUPPLIER_TABLE = 3
Public Const PARTS_TABLE = 4
Public Const SERVICE_TABLE = 5
Public Const EMPLOYEE_TABLE = 6
Public Const EMPLOYEE_PROFILE_TABLE = 7
Public Const COMPANY_PROFILE_TABLE = 8
Public Const INVOICE_TABLE = 9
Public Const INVOICE_DETAILS_TABLE = 10
Public Const PAYMENT_HISTORY_TABLE = 11
Public Const WARRANTY_TABLE = 12
```

## E.8 General File (General.bas)

Option Explicit

```
' constants used for the pre-report dialog
Public Const FINANCIAL_REPORT = 1
Public Const EMPLOYEE_LABOR_REPORT = 2
Public Const SUBLET_REPORT = 3
Public Const SERVICES_REPORT = 4
```

```
Public Const STS_HELP_FILENAME As String = "StsHelp.chm"
```

'Constant help declarations

```
Public Const HH_DISPLAY_TOPIC As Long = &H0
Public Const HH_HELP_CONTEXT As Long = &HF
Public Const HH_TP_HELP_WM_HELP As Long = &H11
Public Const HH_CLOSE_ALL As Long = &H12
```

'HTMLHELP API declarations

```
Public Declare Function HtmlHelp Lib "hhctrl.ocx" Alias "HtmlHelpA" _
    (ByVal hwndCaller As Long, ByVal pszFile As String, _
    ByVal uCommand As Long, ByVal dwData As Long) As Long
```

## E.9 Employee Labor Report (rptEmployeeLabor.dsr)

### Employee Labor (Jack Ripper) From 07/2002 To 12/2002

Month	Labor Hours
7 / 2002	4
8 / 2002	3
9 / 2002	0
10 / 2002	1
11 / 2002	4.75
12 / 2002	0
<b>Total:</b>	12.75

Option Explicit  
Public fTotal As Single  
Public dStartDate As Date  
Public dEndDate As Date  
Public nEmployeeId As Integer

Private Sub ActiveReport\_Initialize()  
' add a button to the toolbar for exporting  
Me.Toolbar.Tools.Add "Export..."  
End Sub

.....

"

" this routine will be called when user clicks the toolbar of report  
" - if it detects that the export button has been clicked the user  
" will be given the option to export the report to a rich text file.  
" - source from the Active Reports Developer's Manual

Private Sub ActiveReport\_ToolbarClick(ByVal Tool As DDActiveReports2.DDTool)  
Dim rtf As New ARExportRTF  
Dim pdf As New ARExportPDF  
Dim sFile As String  
Dim bSave As Boolean

' if our button is pressed...  
If Tool.Caption = "Export..." Then  
frmNoShow.cmdlgeExport.FileName = ""  
frmNoShow.cmdlgeExport.DefaultExt = ".PDF"  
frmNoShow.cmdlgeExport.Filter = "Acrobat (\*.pdf)\*.pdf|Rich Text Format (\*.rtf)\*.rtf"  
frmNoShow.cmdlgeExport.Action = 2 ' open file dialog

If (frmNoShow.cmdlgeExport.FileName <> "") Then ' make sure we got an OK  
' determine whether this is a pdf or rtf file  
If (LCase(Right(frmNoShow.cmdlgeExport.FileName, 3)) = "pdf") Then

```

        pdf.FileName = frmNoShow.cmdnlgExport.FileName ' get new pdf file name
        ' export to the new file
        pdf.Export rptEmployeeLabor.Pages
    ElseIf (LCase(Right(frmNoShow.cmdnlgExport.FileName, 3)) = "rtf") Then
        rtf.FileName = frmNoShow.cmdnlgExport.FileName ' get new rtf file name
        ' export to the new file
        rtf.Export rptEmployeeLabor.Pages
    End If
End If ' end if name was returned from dlg
End If ' end if user clicked 'export' button
End Sub

Private Sub ActiveReport_Activate()
    ' when first called, clear out all counter
    fTotal = 0
End Sub

Private Sub Detail_Format()
    Dim fHours As Single
    Dim nCount As Integer

    ' get the number of hours for this employee on this month
    fHours = EmployeeLaborTotal(nEmployeeId, Val(txtMonth), Val(txtYear))

    ' update label for this row
    lblHours = fHours
    ' increment counter
    fTotal = fTotal + fHours
End Sub

Private Sub PageFooter_Format()
    lblDateTime = Format(Now, "General Date")
End Sub

Private Sub ReportFooter_Format()
    ' update label for the total (only at end of report)
    lblTotal = fTotal
End Sub

Public Function InitLabels(nEmpId As Integer, dStart As Date, dEnd As Date)
    Dim nCount As Integer
    Dim tEmployee As EMPLOYEE

    ' store the values
    dStartDate = dStart
    dEndDate = dEnd
    nEmployeeId = nEmpId
    ' update the labels
    nCount = StsGetRecords("SELECT * FROM EMPLOYEE WHERE EMPLOYEE_ID = " +
Str(nEmpId), EMPLOYEE_TABLE, True)
    If (StsReadEmployee(tEmployee, True)) Then
        lblTitle = "Employee Labor (" + tEmployee.sFirstName + " " + tEmployee.sLastName + ")"
    End If
    lblRange = "From " + Format(dStart, "mm/yyyy") + " To " + Format(dEnd, "mm/yyyy")
End Function

```

## E.10 Financial Report (rptFinance.dsr)

### Financial Report From 07/2002 To 12/2002

Month	Parts	Labor	Sublets	Parts Tax	Labor Tax	Sublets Tax	Total Tax	Total Net	Total Gross
7 / 2002	\$200.00	\$340.00	\$600.00	\$12.00	\$20.40	\$36.00	\$68.40	\$1,140.00	\$1,208.40
8 / 2002	\$268.64	\$204.00	\$650.00	\$16.12	\$12.24	\$39.00	\$67.36	\$1,122.64	\$1,190.00
9 / 2002	\$36.00	\$34.00	\$650.00	\$2.16	\$2.04	\$39.00	\$43.20	\$720.00	\$763.20
10 / 2002	\$17.16	\$68.00	\$939.00	\$1.03	\$4.08	\$56.34	\$61.45	\$1,024.16	\$1,085.61
11 / 2002	\$327.98	\$391.00	\$0.00	\$19.68	\$23.46	\$0.00	\$43.14	\$718.98	\$762.12
12 / 2002	\$54.00	\$109.48	\$543.75	\$3.24	\$6.57	\$32.63	\$42.43	\$707.23	\$749.66
<b>Total:</b>	\$903.78	\$1,146.48	\$3,382.75	\$54.23	\$68.79	\$202.97	\$325.98	\$5,433.01	\$5,758.99

(Source code excluded)



## E.11 Invoice (rptInvoice.dsr)

<b>Heights Service Station</b> Phone: (201)444-6753	11/5/2002 9:56:33 PM	5011
70 Glen Avenue FAX: (201)493-1444	Thank you for your business. Have a great holiday season!	
Midland Park, NJ		

### INVOICE

Flinstone, Fred	Phone: 555-1234	1000 Some OtherCar	Mileage
123 cartoon lane	Work: 555-1234	29	In: 765
Rocks Something, BC	Mobile: 555-1234	WHATEVER	Out: 777

Type	Description	Qty/Hours	Mechanic: Price	Jack Ripper Total
Labor	Tune Up	4.75	\$68.00	\$323.00
Parts	Engine Mounts (Warranty Info: Standard Warranty, 3000 miles and 90 days.)	3	\$76.87	\$230.61
Parts	Transmission Fluid	1	\$13.00	\$13.00
Parts	Air Filter (Warranty Info: Standard Warranty, 3000 miles and 90 days.)	1	\$24.65	\$24.65
Parts	Fuel Filter (Warranty Info: Standard Warranty, 3000 miles and 90 days.)	1	\$50.72	\$50.72

All parts and labor covered for 30 days or 2000 miles.	Parts Total	\$318.98
	Labor Total	\$323.00
	Sub Total	\$641.98
	Tax (.06)	\$38.52
	<b>TOTAL</b>	<b>\$680.50</b>

(Source code excluded)

## E.12 Services/Sublets Report (rptServices.dsr)

### Services Per Month Service: Oil Change From 07/2002 To 12/2002

Month	Amount
7 / 2002	1
8 / 2002	0
9 / 2002	1
10 / 2002	2
11 / 2002	1
12 / 2002	2
<b>Total:</b>	7

(Source code excluded)