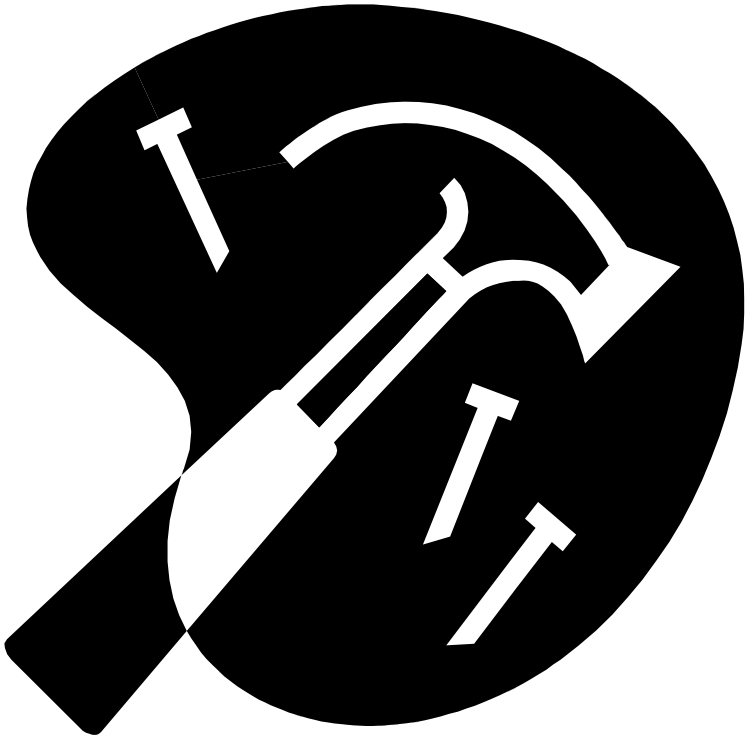
Lime Design

Tool Time Hire System



Adam Esposto
Cassie Martin
David Xalfa
Glenn Eaton
Shawn Reece
Simon Jones

Table of Contents

Introduction

A community based tool hire organisation approached Lime Design which manages a library of tools to the local community

Purpose

As the number of loans and customers is increasing the manual system is required to be replaced by omputer based system. The system will record tool loans and customer details.

Scope

The tool hire organisation in a small retail type environment. Customers approach the clerk at the counter and request the tool they require. The clerk (and assistants from time to time) is responsible for entering all customer and tool data into the database. As tools are borrowed and returned the hires records are updated as required.

The main requirements are:

- Track tools that have been hired
- Provide customers with an invoice at the transaction end.
- Tool scheduling
- Tool availability
- Add, Delete and Edit customers, tools and hires.
- Online help facility
- Reports to inlcude

Chosen Methodology Report

The object oriented methodology is most appropriate for our system for several key reasons.

Object Oriented analysis will allow us greater flexibility and scope in analysing our system. The broader range of tools available within the Object Oriented analysis sphere greatly increases our ability to breakdown the system into understandable chunks and then base our development upon this.

The Object Oriented analysis approach feels more intuitive to use. As humans think of the world in terms of tangible objects, this methodology allows us to break the problem down into specific objects that are much easier to visualise and therefore analyse. Also, as the

boundaries between phases in the development life-cycle are less well defined this leads us to a more iterative approach in our system development. This is also more intuitive as the feeling of continuing to build and add complexity is maintained, allowing a deeper knowledge and understanding based on a firm foundation.

Our initial analysis of our project brief shows us that the project we are to undertake revolves around many different types of objects. The object oriented approach, being as its name suggests, oriented on objects, places emphasis on the objects themselves within a system rather than the process. The actions that are to be performed in our system stem from the difference in the objects that are related to the system. It therefore seems appropriate that placing them at the centre of our development strategy is important. The object oriented approach to system development makes use of modularity, breaking down into classes. These classes, and the methods contained therein, make it easier to reuse code. This enables larger projects to be created in a smaller amount of time and also makes the coding process more efficient. Because of the tight time constraints set upon our minor project, we feel that this ability to re-use code, and the improved efficiency will prove useful in helping us to deliver our project on time.

Modularity also lends itself to our group based project as it enables us to easily divide programming tasks up amongst the groups members. With each person being given a set of required inputs and outputs for their module the need for code to be exactly the same is removed as each module function is self-contained. The data hiding aspects of object oriented ensure that as long as each persons module meets the required specifications for input and output, the code used to achieve the result is not important as there is no possibility of the code being accidentally called at some point in the program.

In summary, the object oriented approach is more appropriate for our system as it will enable us to better analyse the system requirements through the use of more intuitive and iterative development tools, make the best use of our development time by making use of the modularity and re-usability of code, and also enable us to efficiently and effectively work as a team in our coding projects by making further use of object-oriented's modularity and data hiding mechanisms.

Step by Step Activities for Each Phase

Project Planning

Activity	Individual/Group Role	Deliverables/QA Standards	Tools/Techniques
Define the problem	All, esp. Standards Manager	Definition of problem, plus scope	
Produce the project schedule	Team Leader	Project Schedule	
Confirm project feasibility	Standards Manager, Team Leader	contained in System Scope Document (p87)	
Staff the project	Team Leader	Organisational Structure	
Launch the project.	Team Leader	contained in SRS	

(John W. Satzinger, 2009, pp. 40, 45-48)

Analysis

Gather information.	Standards Manager	Research about problem.	
Define system requirements.	Standards Manager	System Requirement Statement	
Build prototypes for discovery of requirements.	Testing Manager	Prototype forms	
Prioritise requirements.	Team Leader	Priority List	
Generate and evaluate alternatives.	Standards Manager, Team Leader.	Evaluation Report	
Review Recommendations with management (project sponsor).	Team Leader, All.	Recommendation report, presented at meeting with management.	

Design

Design and integrate with existing infrastructure.	Standards and Testing Manager		
Design the application architecture.	Testing Manager		
Design the user interfaces (VB forms).	Testing Manager		
Design and integrate the database.	Testing Manager		
Prototype the design details.	Testing Manager		
Design and integrate the system controls.	Testing Manager		

Implementation

Construct the software components.			
Verify and test.	Testing Manager		
Convert Data.			
Train users and document the system.	Admin Manager		
Install the system.			

Support

This area is beyond the scope of the Minor Project. In an industry situation this is an important requirement as companies do not want to invest in a project that will not be well supported once it has been signed off as complete.

For a system in the business world the following activities are required for support.

- Maintain the system.
- Enhance the system.
- Support the users.

Analysis

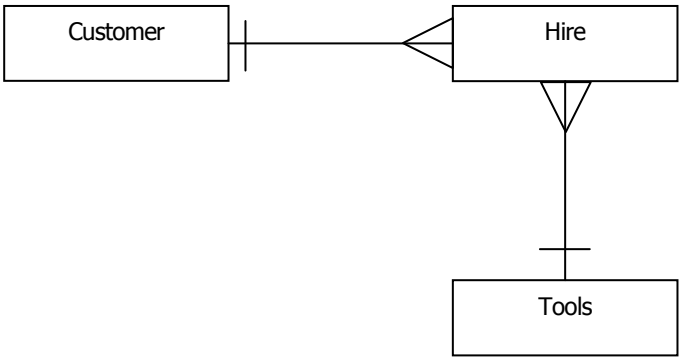
Assumptions Made

In the absence of consultation with the client the following assumptions were made:

- Neither Clerk nor Customers will know toolID's (toolID is simply a field used by the system)
- Tools are never broken, ie if they are not on loan they are available.
- Tool addition date (when the tool was added to the system) isn't included because the tool value never depreciates or is in need of repairs
- Tools are never broken nor do they depreciate, therefore tools don't need to be replaced/ deleted.
- Tools that are returned cannot be hired until the next day.

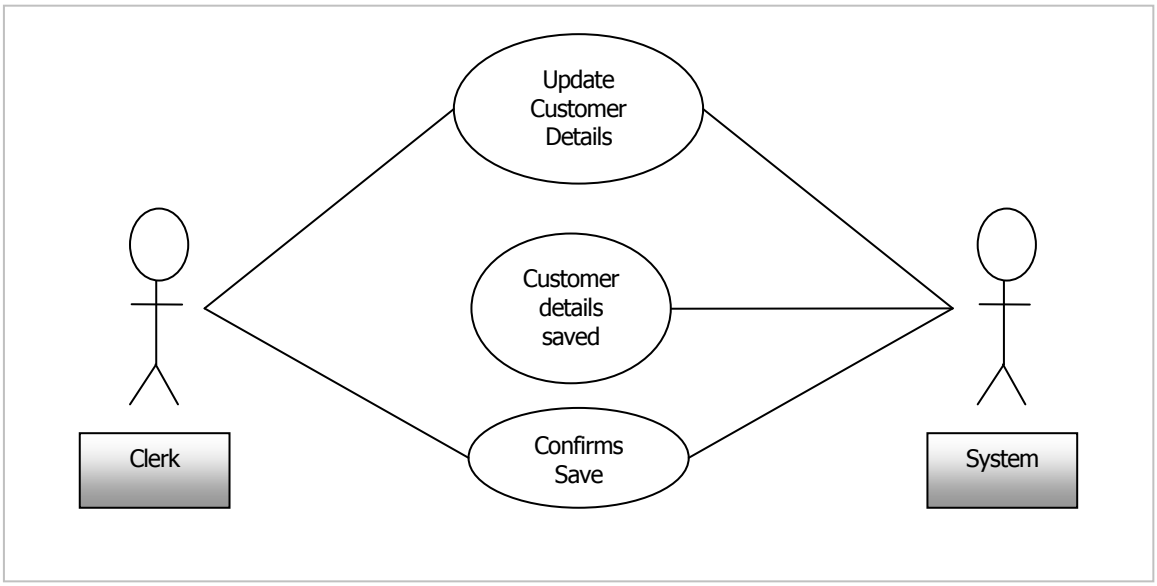
- A tool's maximum hire price is $10,000.
- Customer can hire more than one tool per transaction (invoice)
- Deposit is system wide (80 percent of invoice total)
- Prices are stored per individual tool, therefore each tool can have its own hire rate.
- All tools in the database are in working order and available for hire.
- Customers will only make one transaction per day. (To permit more than one transaction/day another table is required).
- The hire term is exactly one month or one year. (To simplify logic)
- Customers return tools on time.

Entity Relationship Diagram

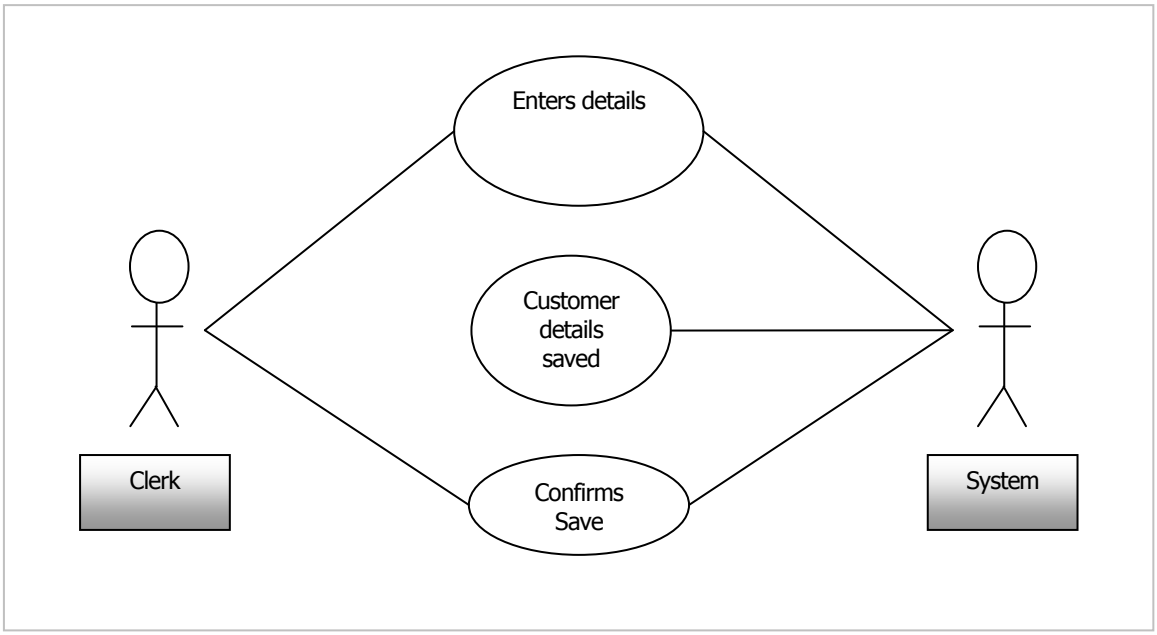


Use Case Diagrams

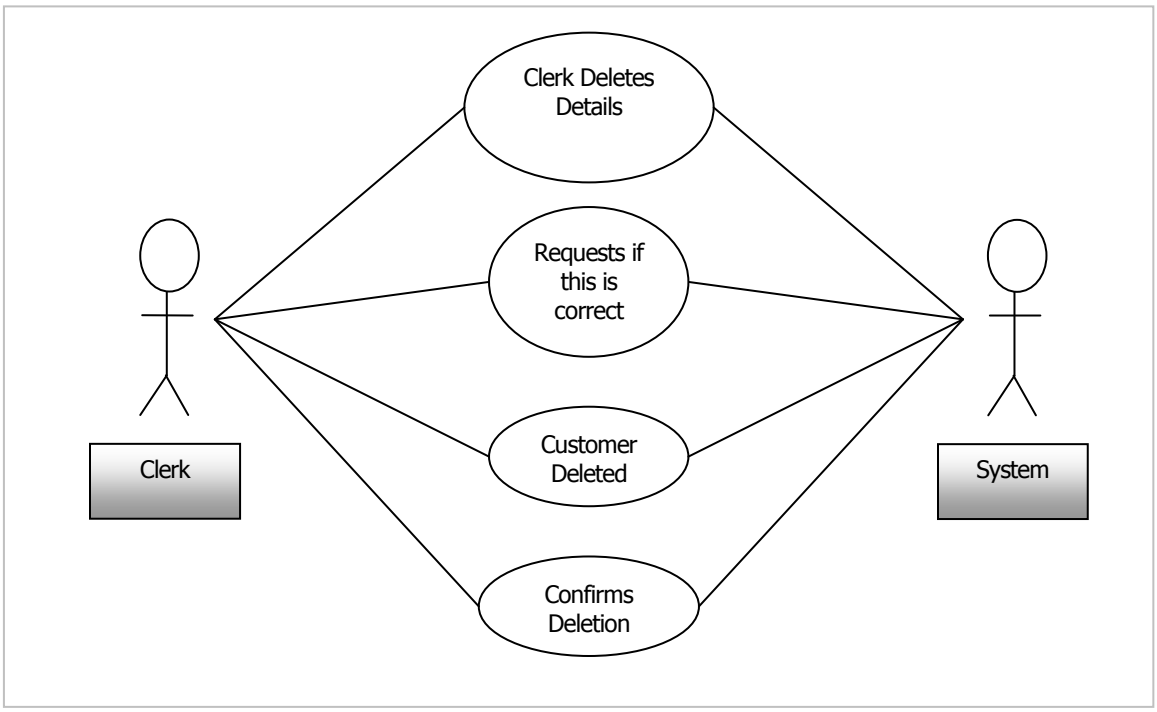
Update Customer



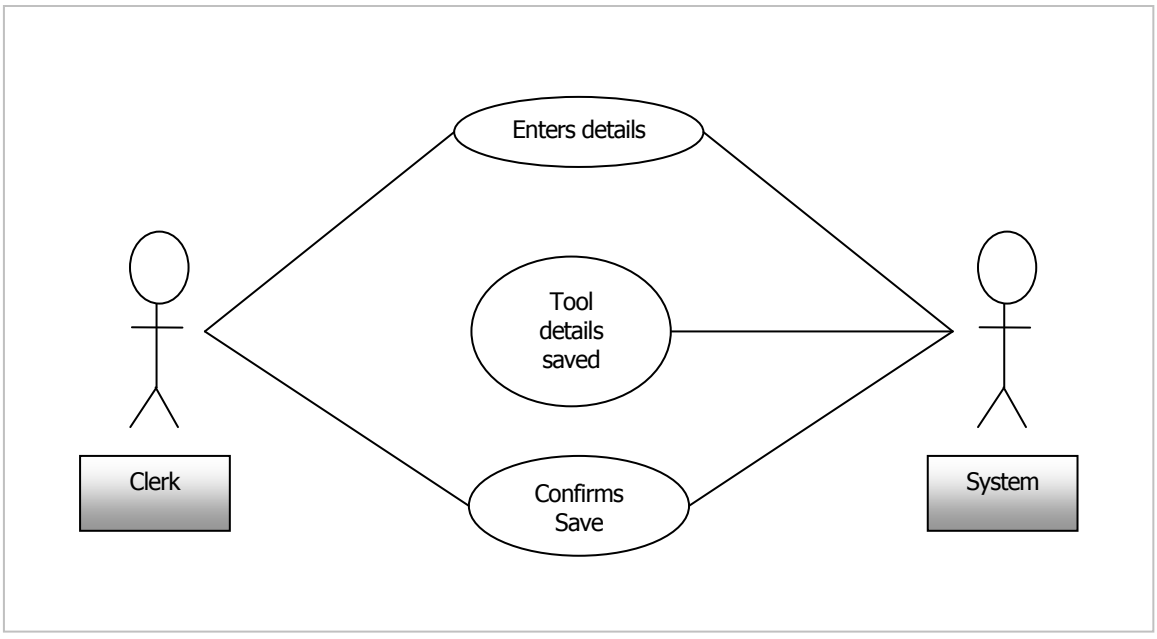
New Customer



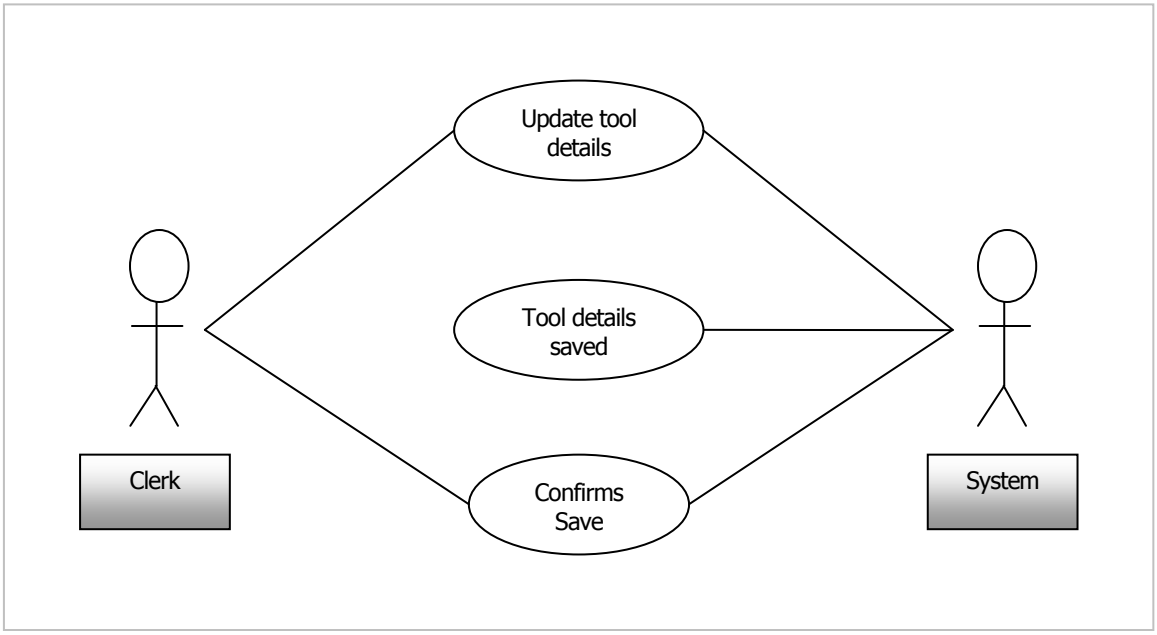
Delete Customer



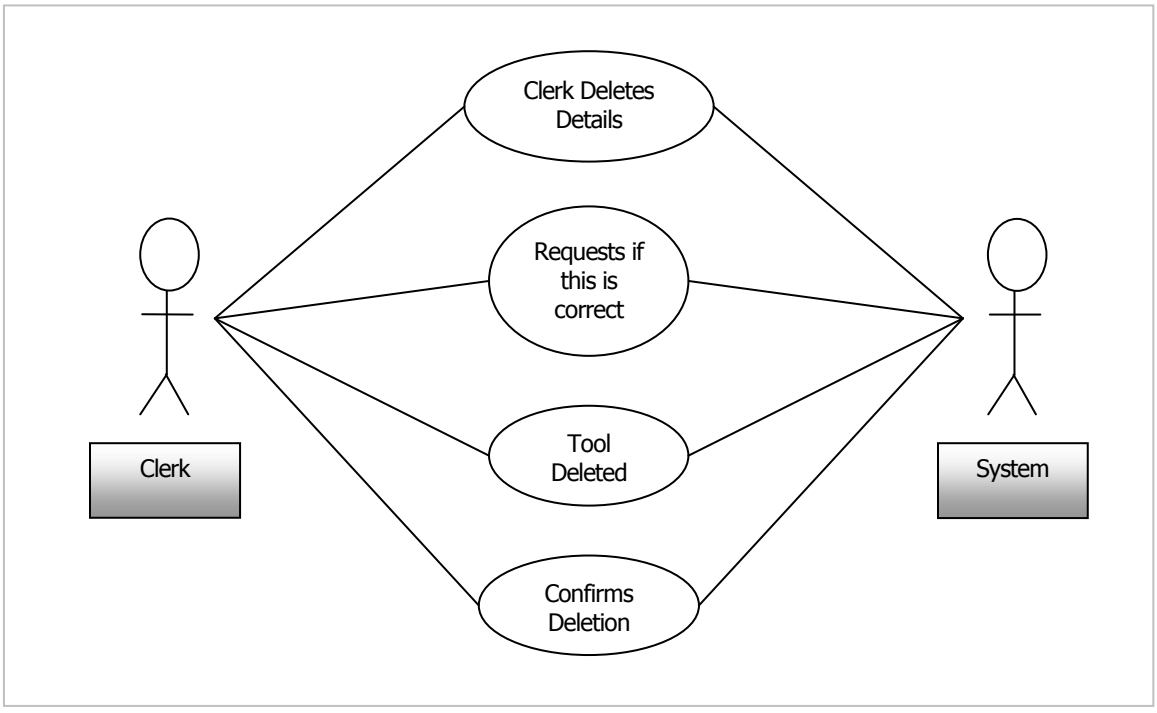
New Tool



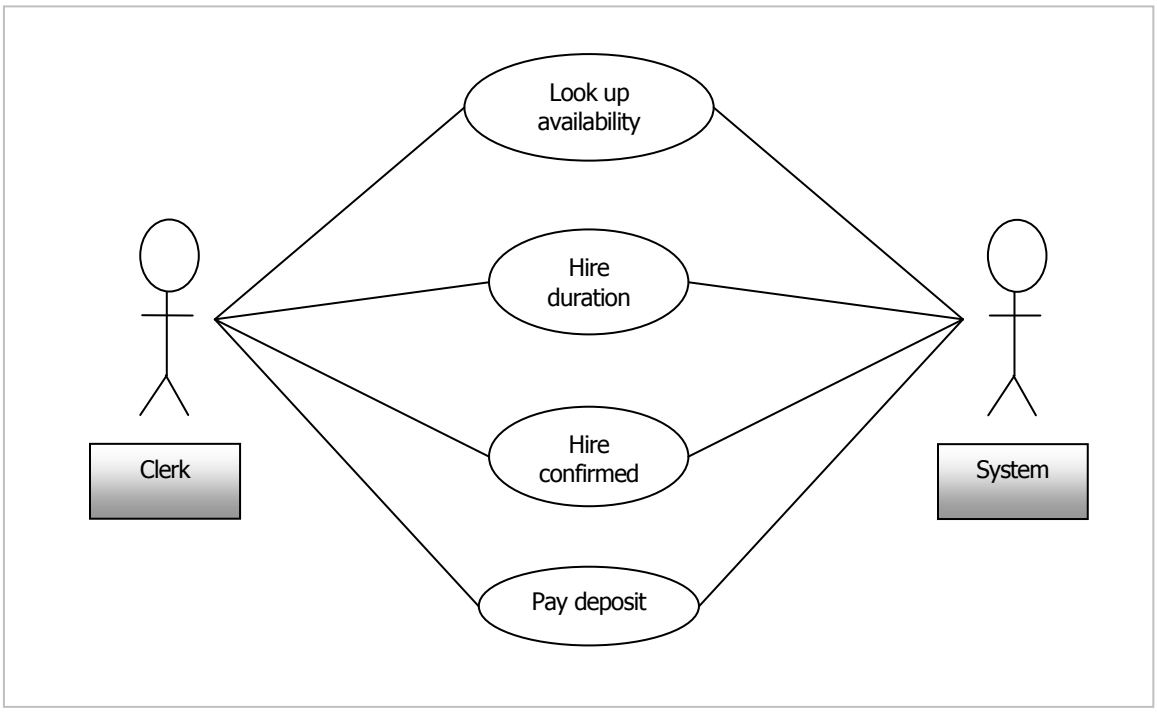
Update Tool



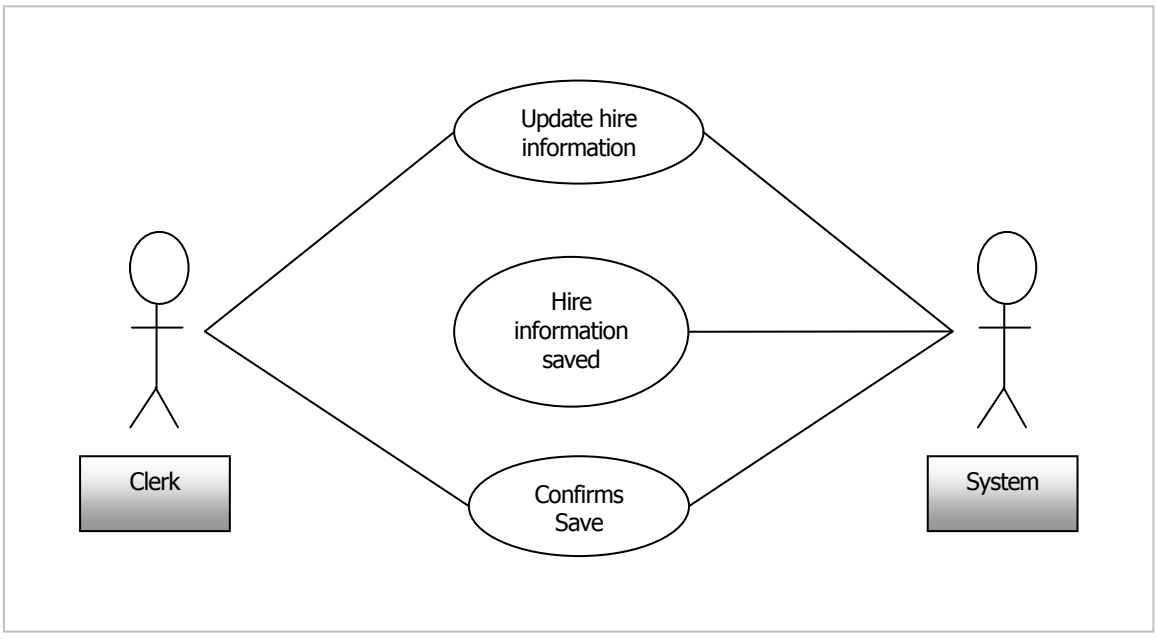
Delete Tool



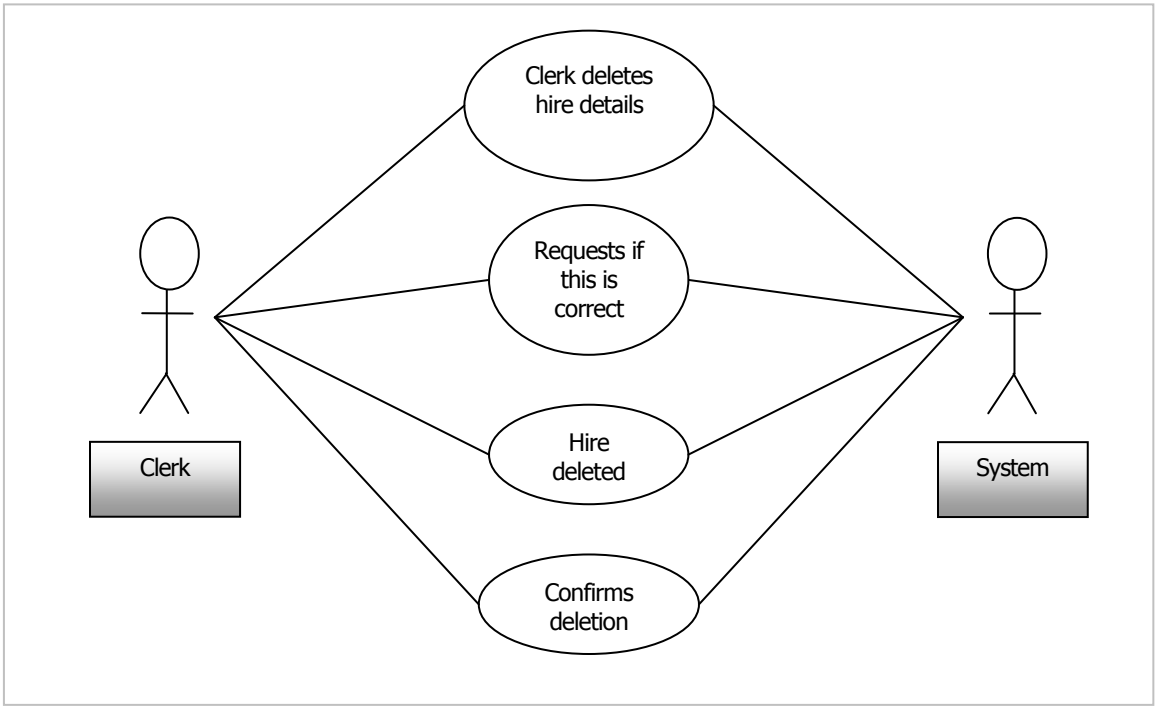
New Hire



Update Hire



Delete Hire



Use Case Descriptions

CreateCustomer

Use Case Name:	CreateCustomer	
Scenario:	Create New Customer record	
Triggering Event:	Clerk needs to add a new customer	
Brief Description:	Clerk needs to create a new customer record. A form collects and validates the required data and stores the record.	
Actors:	Clerk	
Related Use Cases:	Find customer details	
Stakeholders:	Clerk, Customer	
Preconditions:	Customer record does not exist	

Postconditions:	Customer record created, customer issued with ID.	
Main Success Scenario:	**Actor**	**System**
	1. Clerk selects "Create new customer" from menu.	1.1Form is displayed to clerk on screen prompting for required information.
	2.Clerk requests information from customer and enters it onto the form.	2.1Data is validated as the focus leaves each field.
	3. Clerk presses "Save" button.	3.1 Data is saved to database.
		3.2 System responds with confirmation "Customer has been saved"
Extensions:	2.1A System sends message “invalid customer details”.(go back to 2) 2.1BCustomer exists (after finding same name, address or other criteria)(end). 3.1A Database save fails, System sends error message to actor.(end)	

CreateTool

Use Case Name:	CreateTool	
Scenario:	Create New Tool Record	
Triggering Event:	A new tool arrives and needs to be added to the loan pool.	
Brief Description:	Clerk needs to create a new tool record. A form collects and validates the required data and stores the record.	
Actors:	Clerk	
Related Use Cases:	none	
Stakeholders:	Clerk	
Preconditions:	A new tool has arrived.	
Postconditions:	Tool record created, tool issued with ID.	

Main Success Scenario:	Actor	System
	1.Clerk selects "Create new tool" from menu.	1.1 Form is displayed to clerk on screen prompting for required information.
	2.Clerk reads information from tool invoice and enters relevant information onto the form.	2.1 Data is validated as the focus leaves each field.
	3.Clerk presses "Save" button.	3.1 Data is saved to database.
		3.2 System responds with confirmation "Tool has been saved"
Extensions:	2.1A System sends message “invalid tool details”.(go back to 2) 3.1A Database save fails, System sends error message to actor.(end)	

CreateHire

Use Case Name:	createHire	
Scenario:	Clerk wants to hire out an item to a customer	
Triggering Event	Creating a new hire button clicked	
Brief Description:	customer confirms hire and system saves the hire record.	
Actors:	Clerk	
Related Use Cases:	searchCustomer, searchTool	
Stakeholders:	Clerk, Customer, Hire Tool Owners	
Preconditions:	Customer is registered, tool is found + available	
Postconditions:	A new tool hire is created	
Main Success Scenario:	**Actor**	**System**
	1.Clerk clicks hire button	1.1 Tool Hire (hire record) is saved to database.
		1.2 System responds with confirmation "Hire has been saved"

Extensions:	1.1a Database update fails, System sends error message to actor.(end)

Edit Hire

Use Case Name:	editHire	
Scenario:	Clerk edits hire details	
Triggering Event	Clerk clicks edit	
Brief Description:	clerk clicks edit, clerk modifies details, system saves the hire record.	
Actors:	Clerk	
Related Use Cases:	searchHire	
Stakeholders:	Clerk, Customer, Hire Tool Owners	
Preconditions:	Hire is found, Customer is registered	
Postconditions:	The Hire is updated	
Main Success Scenario:	**Actor**	**System**
	1. Clerk presses "Edit" button.	1.1 Edit mode enabled
	2. Clerk modifies Hire details.	2.1 Data is validated as the focus leaves each field.
	3. Clerk presses "Save" button.	3.1 The database Hire record is updated .
		3.2 System responds with confirmation "Hire has been updated".
Extensions:	2.1A System sends message "data invalid re-enter details".(go back to 2) 3.1A Database update fails, System sends error message to actor.(go back to 1)	

Delete Hire

Use Case Name:	deleteHire	
Scenario:	Clerk deletes hire details	
Triggering Event	Clerk clicks delete hire	

Brief Description:	clerk clicks delete hire, system asks are you sure, clerk confirms, deletes and sends system confirmation message.	
Actors:	Clerk	
Related Use Cases:	searchHire	
Stakeholders:	Clerk, Customer, Hire Tool Owners	
Preconditions:	Hire is found, Customer is registered	
Postconditions:	The Hire is updated	
Main Success Scenario:	**Actor**	**System**
	1.Clerk presses "delete hire" button.	1.1 System asks for confirmation(are you sure message)
	2. Clerk clicks confirm deletion.	2.1 System deletes hire record.
		2.2 System responds with confirmation "Hire has been deleted".
Extensions:	2.A Customer clicks cancel.(end) 2.1A Database row deletion fails, System sends error message to actor.(go back to 1)	

Search Hire

Use Case Name:	searchHire	
Scenario:	Clerk needs to locate a hre Locating of a hire	
Triggering Event	Clerk clicks hire search button	
Brief Description:	Clerk enters search criteria, system retrieves & displays hire details	
Actors:	Clerk	
Related Use Cases:	searchCustomer	
Stakeholders:	Clerk, Customer, Hire Tool Owners	
Preconditions:	customer is registered	
Postconditions:	Hire is found	

Main Success Scenario:	Actor	System
	1.Clerk enters hire search criteria.	1.1Data is validated as the focus leaves each field.
	2. Clerk presses “Search“ button.	2.1 System retrieves and displays hire.
		2.2 System responds with confirmation "Hire has been updated”.
Extensions:	1.1A System sends message “invalid search criteria”.(go back to 1) 1.2A System message “Hire not found” (go back to 1) 2.1A Database query fails, System sends error message to actor.(go back to 1)	

Update Customer

Use Case Name	UpdateCustomer	
Scenario		
Triggering Event	Clerk needs to change Customer details	
Brief Description	Updates the Customers details as held in the system.	
Actors	Clerk	
Related Use Cases	Includes: SearchCustomer()	
Stakeholders	Clerk	
Preconditions	Existing Customer	
Postconditions	Updated Customer	
Main Success Scenario	Actor	System
	1. Input changes to Customer	1.1 Checks changes are valid
		1.2 Writes changes to database
		1.3 Send confirmation to Actor
	2. Receive confirmation	
Extensions	1.2A Invalid Changes: Send error message to Actor, Return to 1. 1.2B Database update fails: send error message to Actor. Return to 1.	

Delete Tool

Use Case Name	DeleteTool	
Scenario		
Triggering Event	Clerk needs to delete a tool from the system	
Brief Description	Delete a tool from the system.	
Actors	Clerk	
Related Use Cases	includes: SearchTool	
Stakeholders	Clerk	
Preconditions	Existing Tool	
Postconditions	Tool deleted	
Main Success Scenario	**Actor**	**System**
	1. Request deletion of tool	
		1.1 Request confirmation from Actor
	2. Confirm deletion	
		2.1 Delete record from database
		2.2 Send confirmation to Actor
	3. Receive confirmation of deletion	
Extensions	2.1A Database update fails: send error message to Actor. Return to 1.	

Delete Customer

Use Case Name	DeleteCustomer()	
Scenario	Delete a Customer Record	
Triggering Event	Clerk deletes a customer	
Brief Description	deletes a customer	
Actors	Clerk	
Related Use Cases	SearchCustomer()	
Stake Holders	Clerk, Customers and hire tool owners	
Preconditions	needs existing customers	
Post Conditions	Customer deleted	
Main Success	**Actor**	**System**

	1.Request Deletion of a Customer	1.1 Request confirmation from Clerk
		Searches for Customer
	2. Confirm Deletion	2.1 Delete Record form Database
		2.2 Send confirmation to clerk
	3. Receive Confirmation of Deletion If Changes are invaild return to 1	
Extensions:	Database update fails: send error message to Actor. Return to 1.	

Update Tool

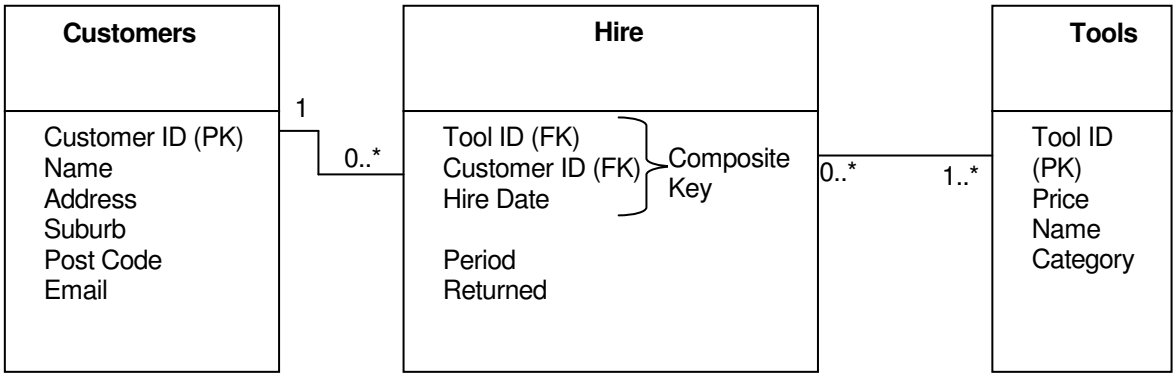
Use Case Name	UpdateTool()	
Scenario	Updating of a tool	
Triggering Event	Clerk Updates a tool	
Brief Description	Adds or changes tools	
Actors	Clerk	
Related Use Cases	SearchTool()	
Stake Holders	Clerk, Customers and hire tool owners	
Preconditions	Existing tool	
Post Conditions	Update of Tool	
Main Success	**Actor**	**System**
	1. Clerk changes tools details	1.1 System then checks if changes are valid
		1.2 System then updates the changes to the database
		1.3 Sends confirmation to clerk
	2. clerk receives confirmation	
Extensions	2.1A Database update fails: send error message to Actor. Return to 1.	

Event Table

Event	Trigger	Source	Use Case	Response	Destination
1. Customer Registers with Tool	Register	Customer	New customer	Saved Customer(CustID)	Customer

Time.					
2. Customer makes a hire	Hire	Customer	New Hire	Hire	Hire
3. Tool Time purchases a new tool.	New Tool	Tool	New Tool	Saved tool	Tool

Class Diagram



Hardware and Software

The software used for the ToolTime project consists of the following:

- Windows XP Professional Operating System
- Microsoft Office 2007
- Microsoft Access 2007
- Microsoft Visual Studio 2005

The hardware used for this application was provided in room EA221 as this is where the application was built. No network was built for this application as they are using a stand alone.

Naming Conventions

General Naming Guidelines:

Names that are given should be descriptive and have titled what is being used e.g txtFirstName (A textbox on a form which shows a first name input textbox). Names that require multiple words should be capitalised with its first letter of every new word used. (camel case text). e.g. txtFirstName.

Form Elements

Element	Prefix	Example
Button	btn	btnResult
CheckBox	chk	chkMale

Element	Prefix	Example
ComboBox	cbo	cboEnglish
DataRow	dr	drTool
DataSet	ds	dsToolTime
DataTable	dt	dtCustomer
Form	frm	frmOfficeSuppliers
GroupBox	grp	grpDetails
Label	lbl	lblFirstName
ListBox	lst	lstPolicyCodes
MenuScript	mnu	mnuFile
RadioButton	rdo	rdoFemale
SaveFileDialog	sfd	sfdSaveInvoice
Child MDI form	mdi	mdiCreateOrder

Variable Naming

Element	Prefix	Example
Boolean	bln	blnIsValid
date	dtm	dtmDateHired
Double, Single, Currency, Decimal	dbl	dblCost
Integer, Long (Integer)	int	int
Object	obj	objNewOrder
String	str	strCustomerFirstName
TextBox	txt	txtFirstName
Timer	tmr	tmrTimer

(Microsoft, 2003) (Crowley, 2001)

Data Dictionary

Table	Description	Field Name	Type	Sample	Null?	References	PK/FK
Customers	Unique identifier for each customer	CustID	VarChar(8)	CUS00001	NN	-	PK
	First name of customer	firstName	VarChar(100)	Bob	NN	-	-
	Last name of customer	lastName	VarChar(100)	Smith	NN		
	House number and street name of customer	Addr	VarChar(255)	43 Wattle Grove	NN	-	-
	Suburb of customer	Suburb	VarChar(100)	Treeview	NN	-	-
	Post Code of customer	PostCode	VarChar(4)	2012	NN	-	-
	Phone Number of customer	PhoneNum	VarChar(10)	0395261112	NN	-	-
	E-mail of customer	Email	VarChar(255)	bob.smith@wattle.com.au		-	-
Hire	Unique identifier for tool hired	TID	Varchar(8)	TOO03000	NN	Tool	FK
	Date of hire	HireDate	Date	2009-05-02	NN	-	PK
	Unique identifier for customer hiring tool	CustID	VarChar(8)	CUS00001	NN	Customer	FK
	Period of hire, monthly or yearly	Period	Char	M	NN	-	-
	Has the tool been returned?	Returned	Yes/No	TRUE	NN	-	-

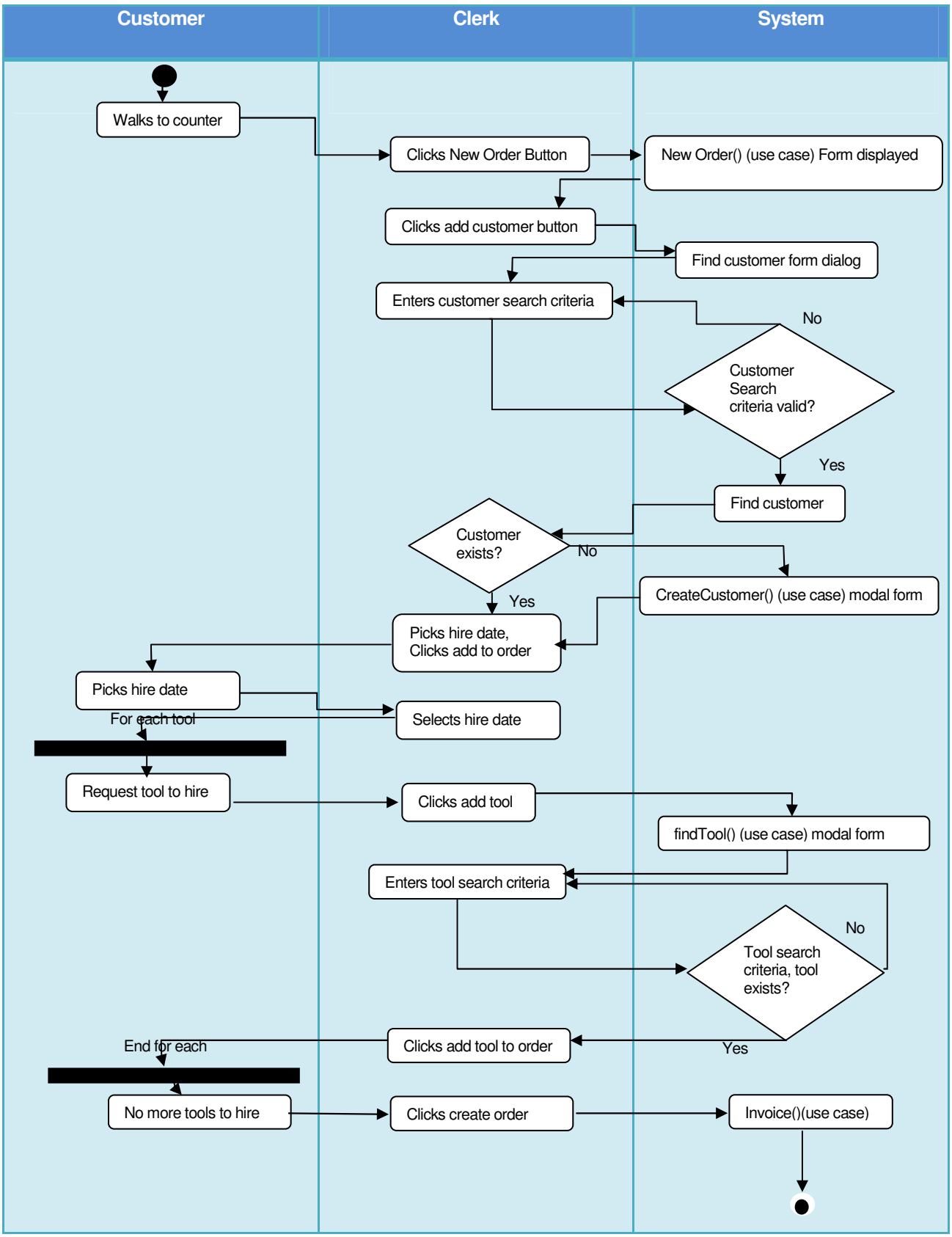
Table	Description	Field Name	Type	Sample	Null?	References	PK/FK
Tool	Unique identifier for each tool	TID	Varchar(8)	TOO03000	NN	-	PK
	Tool hire price	Price	Currency	$52.00	NN	-	-
	Tool Name	Name	VarChar(50)	Hammer	NN	-	-
	Tool category	Category	VarChar(20)	Power Tools	NN	-	-

Database Setup SQL

The following SQL has been written for Microsoft Access. Modifications are requred if the script is to be used for other DBM Systems

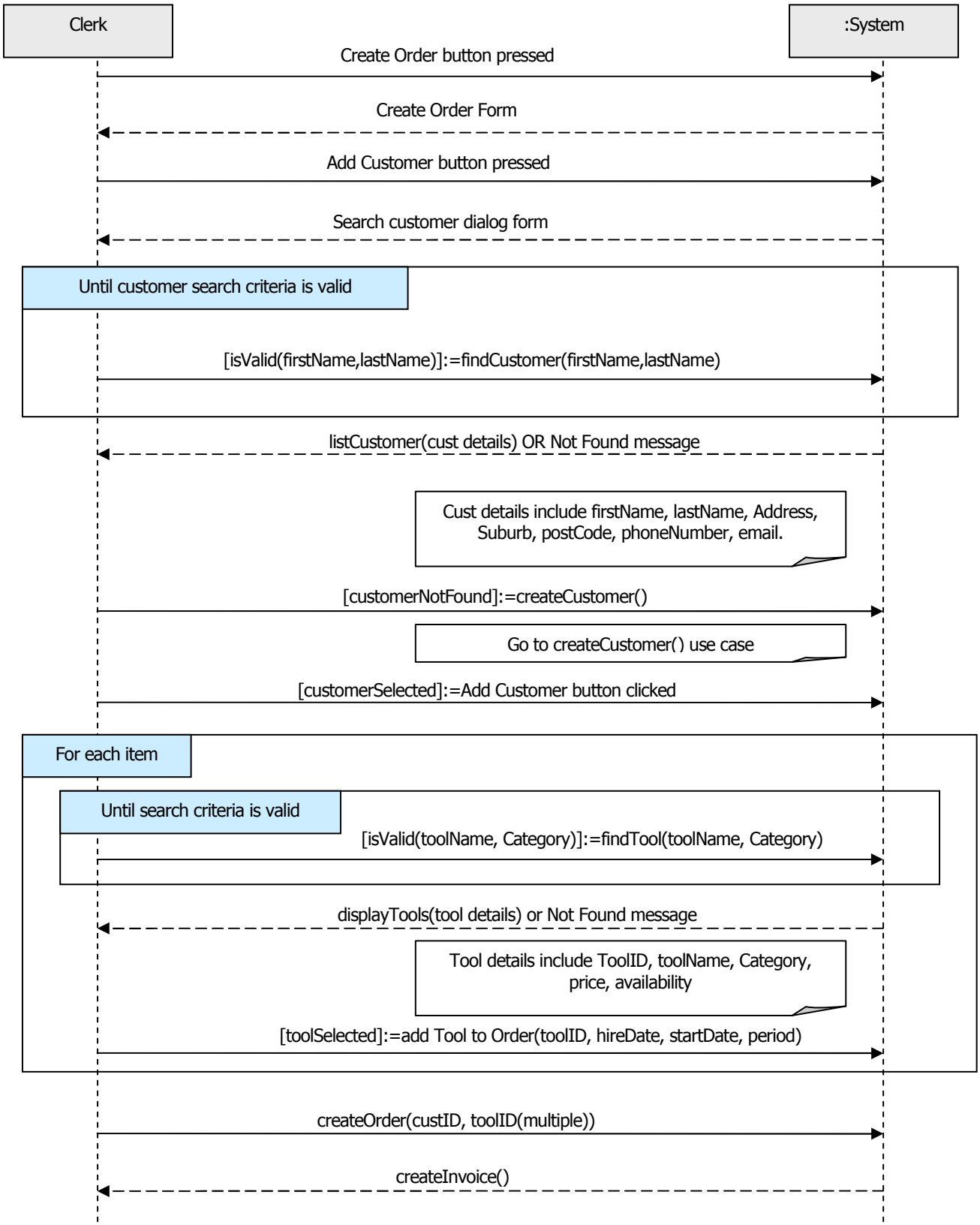
```
CREATE TABLE Customer
(custID varchar (8) CONSTRAINT pkey PRIMARY KEY,
fName varchar (50) NOT NULL,
lName varchar (50) NOT NULL,
Address varchar(255) NOT NULL,
Suburb varchar(100) NOT NULL,
postCode varchar(4) NOT NULL,
phoneNum Varchar(10) NOT NULL,
Email varchar (255));

CREATE TABLE Tool
(tID varchar (8) CONSTRAINT pkey PRIMARY KEY,
Price currency NOT NULL,
Name varchar(50) NOT NULL,
Category varchar(20) NOT NULL);

CREATE TABLE Hire
(tID varchar(8) CONSTRAINT TIDFK REFERENCES Tool,
hireDate Date NOT NULL,
custID varchar (8) CONSTRAINT CustIDFK REFERENCES Customer,
Period Char NOT NULL,
Returned BIT NOT NULL);

CREATE INDEX hireIndex
ON Hire (hireDate, tID, custID)
WITH PRIMARY;
```

Activity Diagram



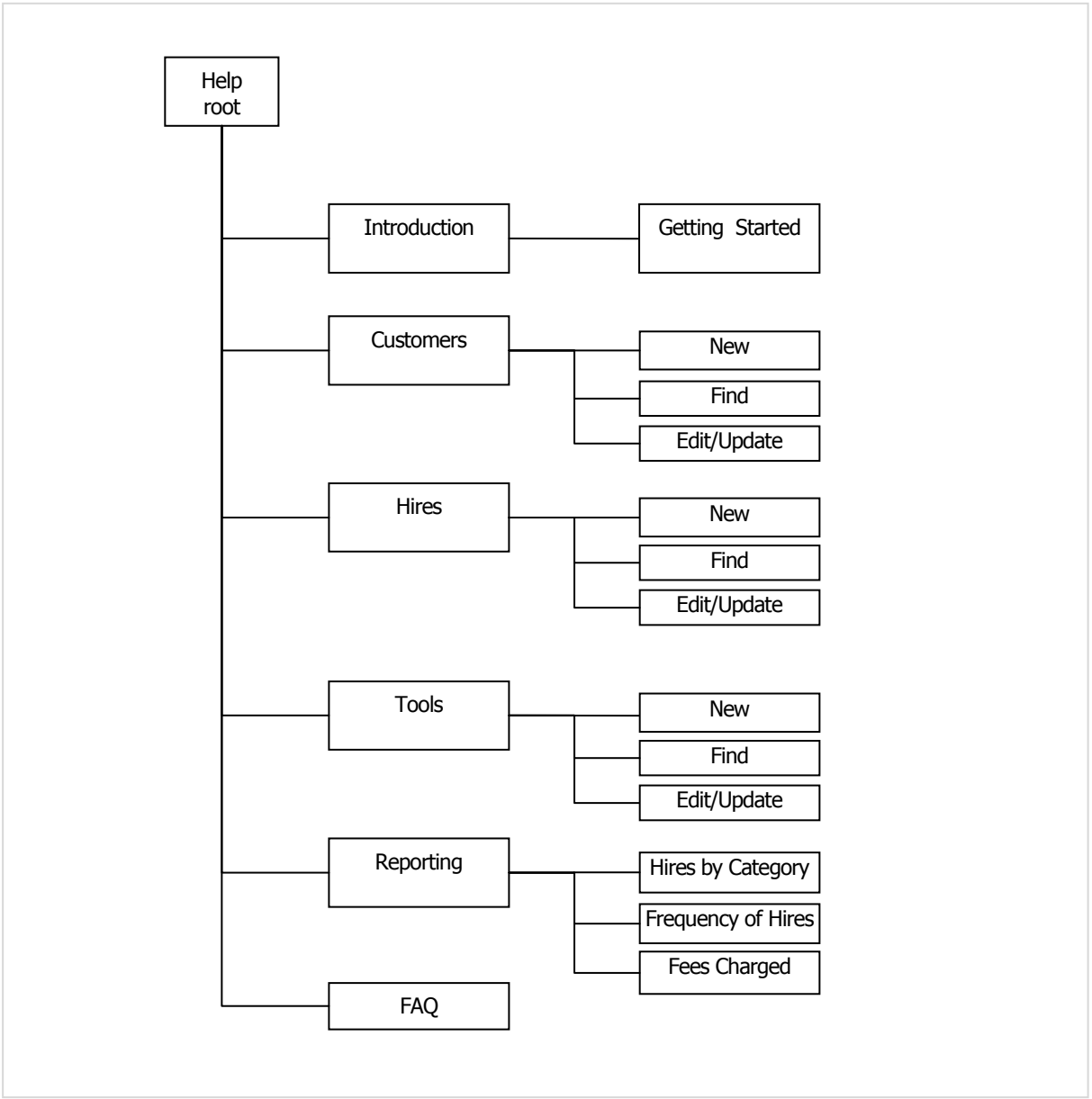
Sequence Diagrams

Create New Order



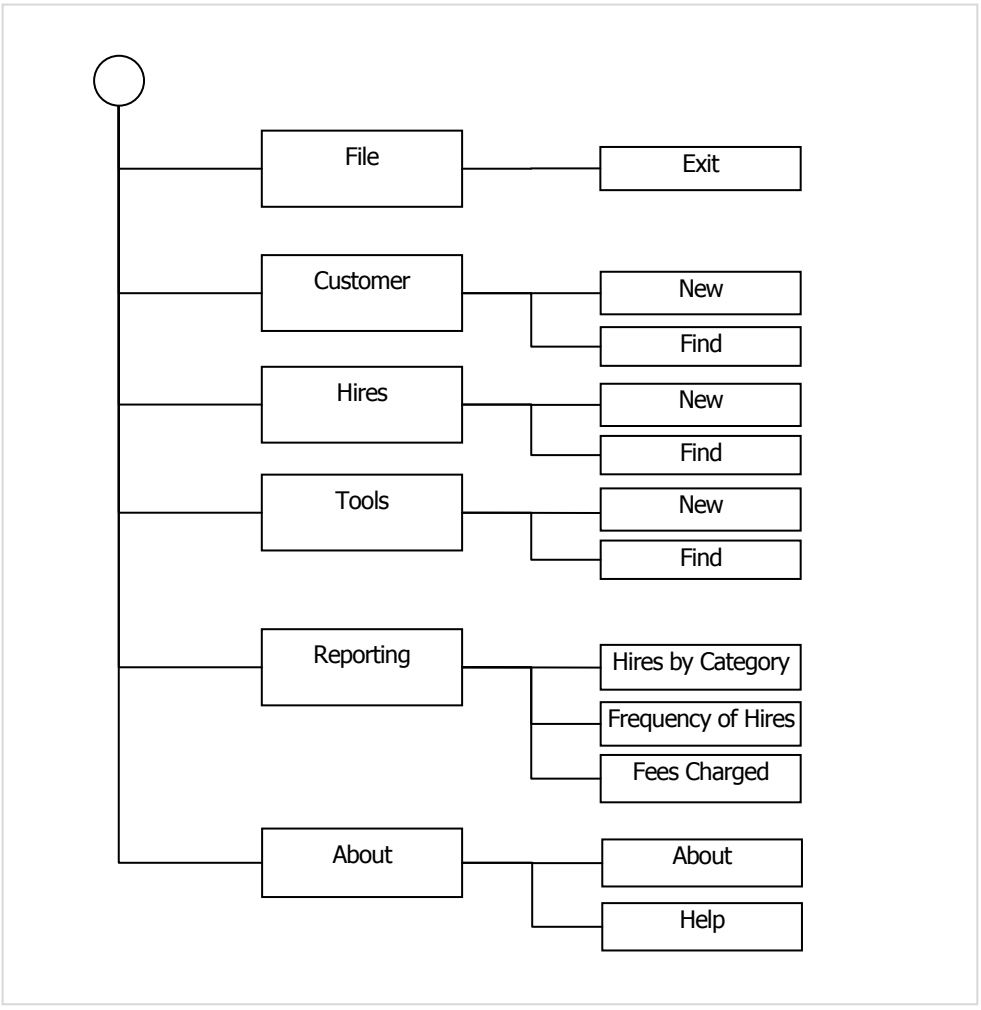
Online Help Menu

The compiled help file (chm format) was created using Microsoft's HTML Help Workshop, using the following topic structure. The source files are located in fhe folder *OnlineHelp-source* on the Tool Time CD-ROM.



Tooltime Top Menu Structure

The diagram below is the menu tree structure. This menu is located at the top of the application's main screen.

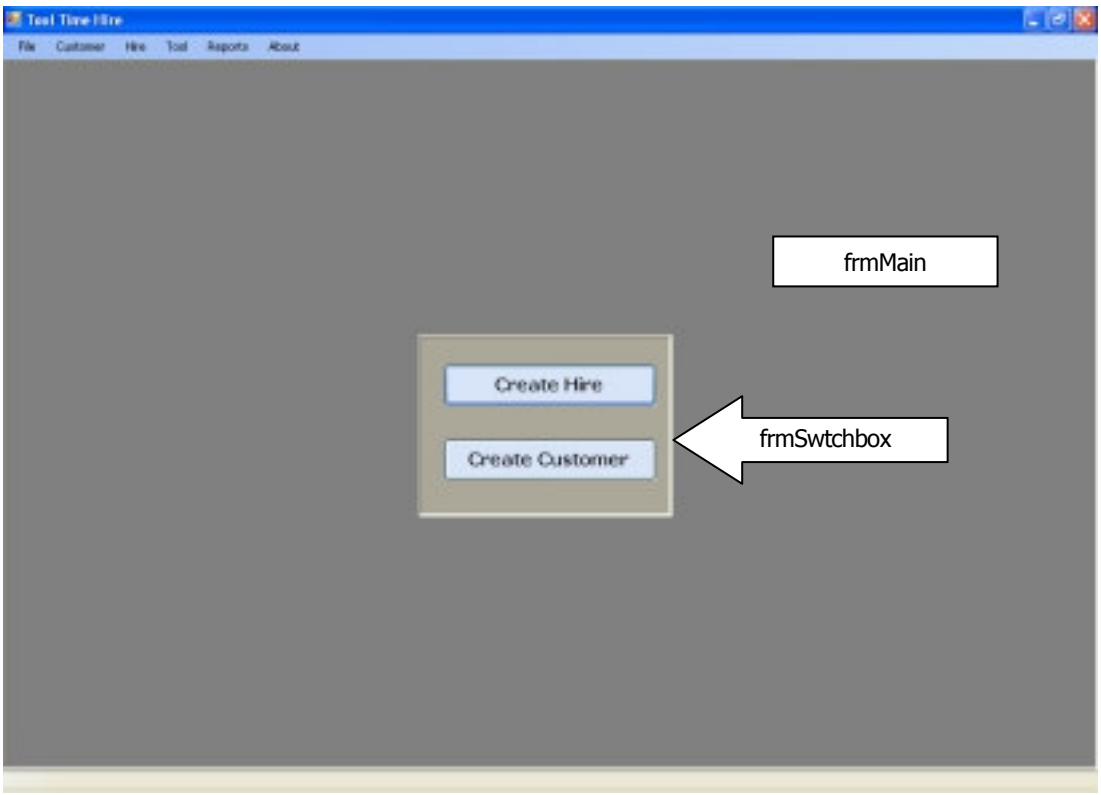


Software Modules and Forms

This section contains all of the forms and an overall description of code behind.

Please note the forms are shown at different sizes. The larger ones have been reduced in size to fit the document.

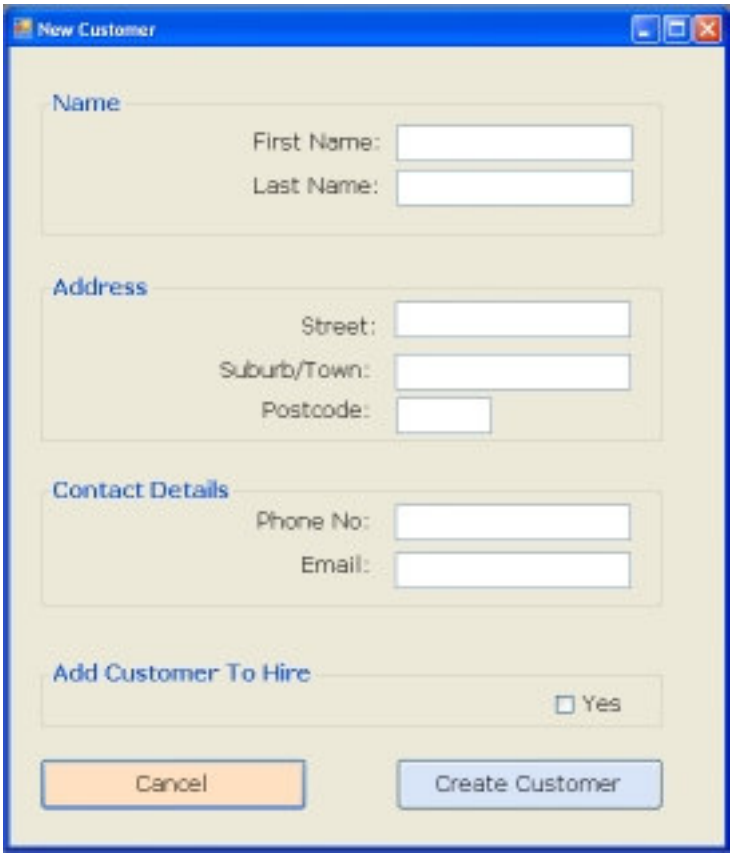
Main Screen



Form	frmMain

Form	mdiSwitchBox
Buttons	btnCreateOrder btnCreateCustomer

New Customer



Form Components

Form	mdiNewCustomer
Buttons	chkAddCustHire btnCancel btnCreateCustomer
Labels	lblFirstName lnlLastName lblStreet lblTown lblPostCode lblPhoneNo lblEmail
Text Boxes	txtFirstName txtLastName txtStreet txtTown txtPostCode txtPhoneNo txtEmail

Functional Procedures

Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnCreateCustomer	Update database
btnCancel	Cancel update Close form

Object variables

Scope	Name and Type
Global	dataClass As dataClass dataClassSet As ToolHireDataSet custBindingSource As BindingSource strCustID As String
Local	msbxAns As DialogResult Dim dateAndTimeDate As Date Dim strMessage As String Dim deleteDialogResult As DialogResult intCustID As Integer

Edit Customer

Form Components

Form	mdiEditCustomer

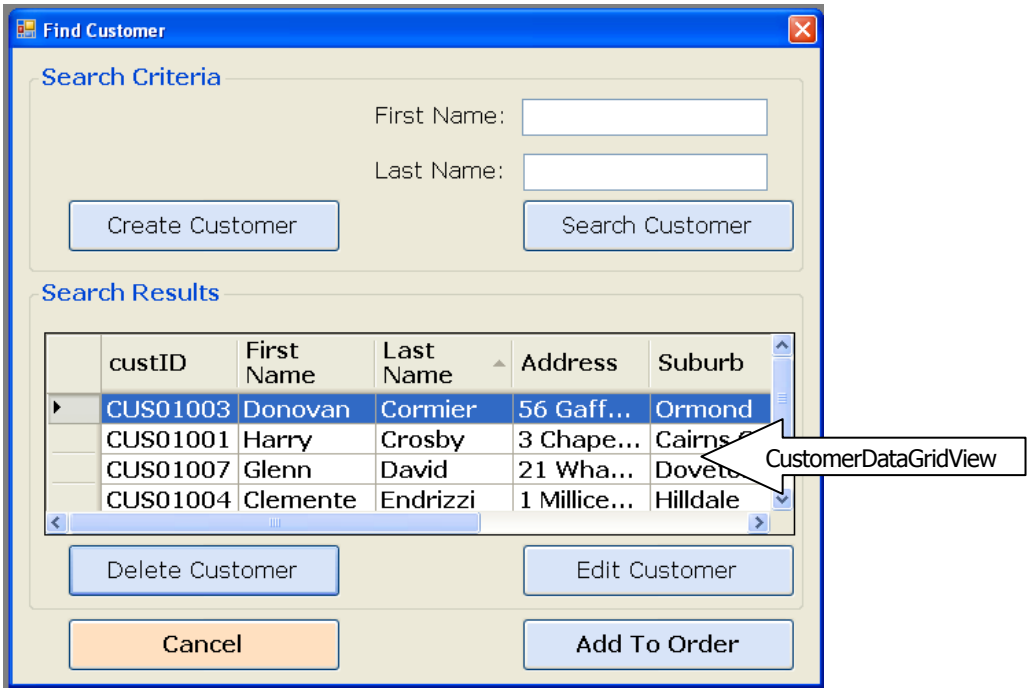
Buttons	chkAddCustHire btnCancel btnUpdateCustomer
Labels	lblFirstName lnlLastName lblStreet lblTown lblPostCode lblPhoneNo lblEmail
Text Boxes	txtFirstName txtLastName txtStreet txtTown txtPostCode txtPhoneNo txtEmail

Functional Procedures

Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnUpdateCustomer	Update database
btnCancel	Cancel update Close form

Object variables

Scope	Name and Type
Global	dataClass As dataClass dataClassSet As ToolHireDataSet custBindingSource As BindingSource strCustID As String Private Shared anInstance As mdiEditCustomer
Local	



Find Customer

Form Components

Form	mdiFindCustomer
Buttons	btnCancel btnEditCustomer btnCreateCustomer btnDeleteCustomer btnSearchCustomer btnAddToOrderr
Labels	lblFirstName lblLastName
Text Boxes	txtFirstName txtLastName lstSearchResults

Functional Procedures

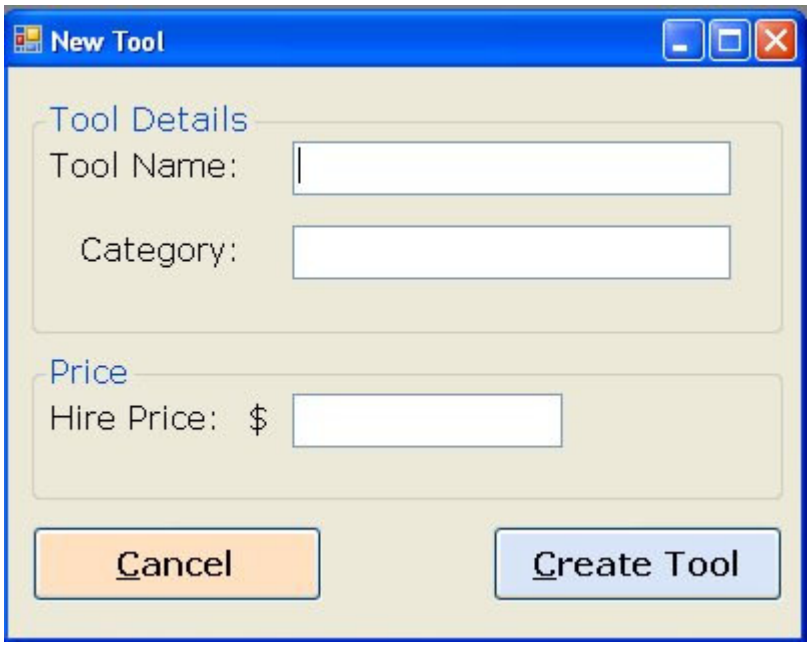
Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnEditCustomer	mdiEditCustomer
btnCreateCustomer	mdiNewCustomer
btnDeleteCustomer	Delete Customer

btnSearchCustomer	Search for customer
btnAddToOrderr	Add customer to Data Grid View
btnCancel	Cancel update Close form

Object variables

Scope	Name and Type
Global	Private Shared anInstance As mdiFindCustomer dt As New DataTable rowIndex As Integer = 0 mode As String Private taCustomer As ToolHireDataSetTableAdapters.CustomerTableAdapter myData As dataClass aDataSet As ToolHireDataSet bsCustomer As BindingSource
Local	mdiEditCustomer As New mdiEditCustomer customerName As String fName As String lName As String

New Tool



Form Components

Form	mdiNewTool
Buttons	btnCancel btnCreateTool
Labels	lhlToolName

	lblCategory lblHirePrice
Text Boxes	txtToolName txtCategory txtHirePrice

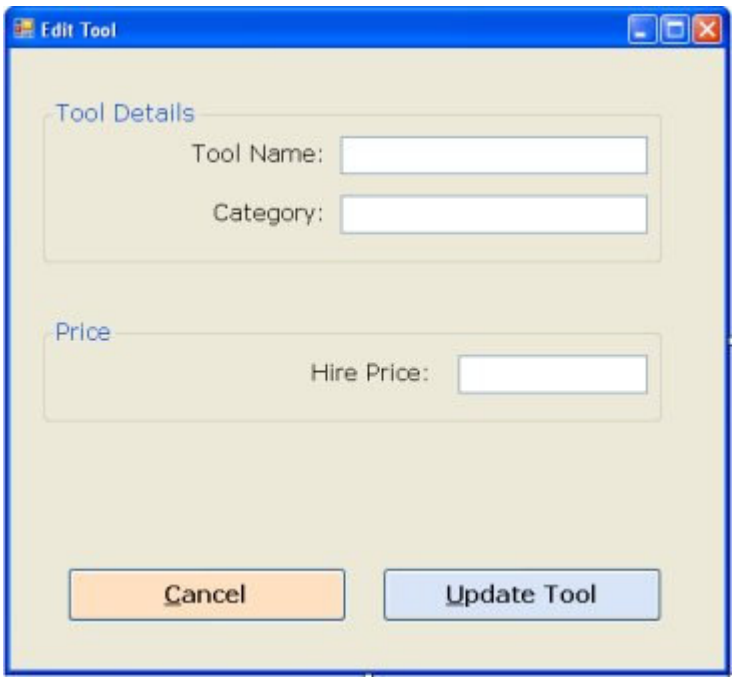
Functional Procedures

Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnCreateTool	Update database
btnCancel	Cancel update Close form

Object variables

Scope	Name and Type
Global	Private Shared anInstance As mdiNewTool Private dataClass As dataClass Private aDataSet As ToolHireDataSet Private toolTable As ToolHireDataSet.ToolDataTable Private toolRow As ToolHireDataSet.ToolRow
Local	dblHirePrice As Double strToolName, strCategory, tID As String dlgConfirm As DialogResult toolNumber As Integer dlgResult As DialogResult stringPattern As String

Edit Tool



Form Components

Form	mdiEditTool
Buttons	btnCancel btnUpdateTool
Labels	lhlToolName lblCategory lblHirePrice
Text Boxes	txtToolName txtCategory txtHirePrice

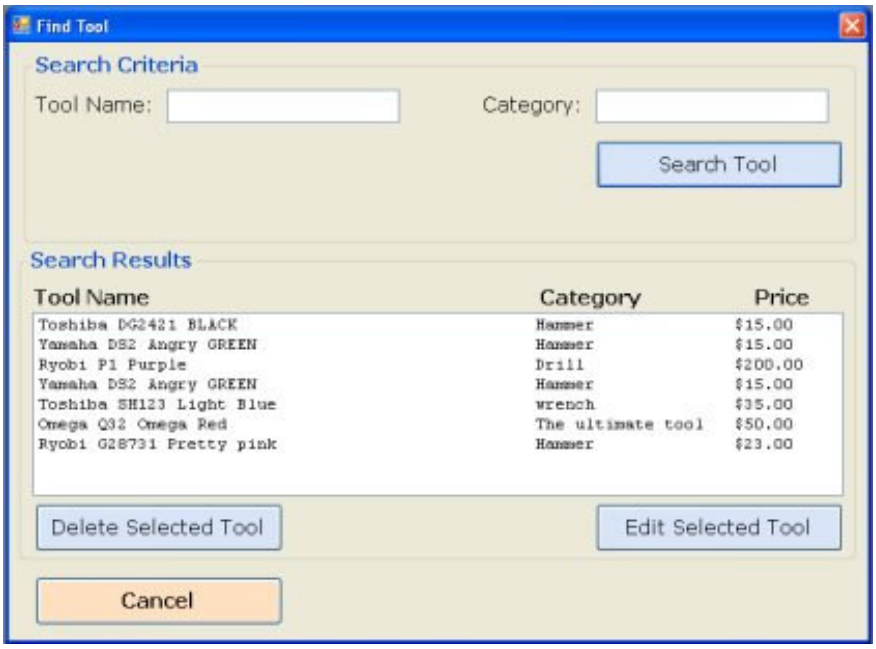
Functional Procedures

Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnUpdateTool	Update database
btnCancel	Cancel update Close form

Object variables

Scope	Name and Type
Global	Private Shared tID As String Private Shared anInstance As mdiEditTool
Local	

Find Tool



Form Components

Form	mdiFindTool
Buttons	btnSearchTool btnRemoveSelTool btnDeleteSelected btnEditSelected btnAddToHire btnCancel
Labels	lhlToolName lblCategory lblHirePrice
Text Boxes	txtToolName txtCategory lstSearchResults

Functional Procedures

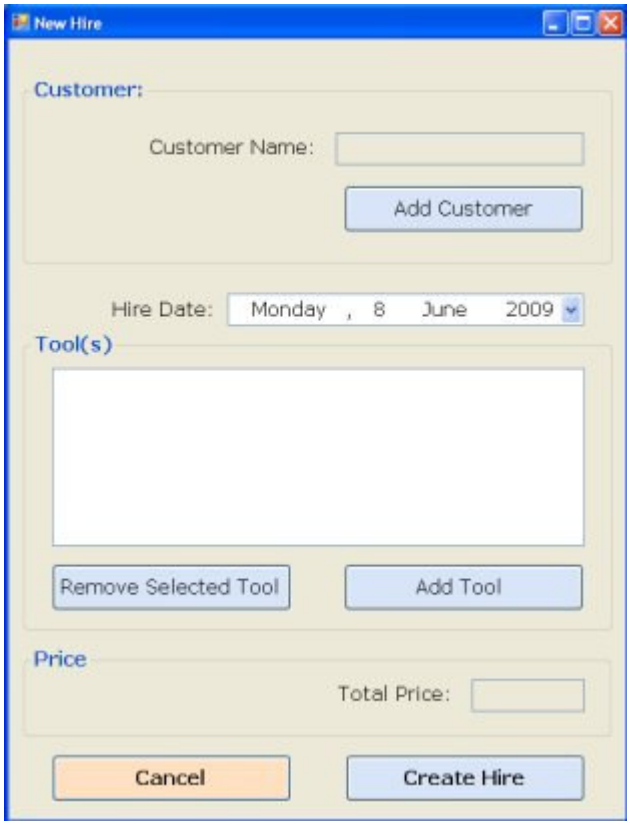
Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnSearchTool	Search for Tool
btnRemoveSelTool	Remove tool from list
btnDeleteSelected	Delete selected tool from database
btnEditSelected	mdiEditTool with toolID of selected tool

btnAddToHire	Add tool to Hire
btnCancel	Close form

Object variables

Scope	Name and Type
Global	Private dataClass As dataClass Private businessClass As businessClass Private aDataSet As ToolHireDataSet Private toolTable As ToolHireDataSet.ToolDataTable Private toolRow As ToolHireDataSet.ToolRow Private Shared anInstance As mdiFindTool Private Shared strDirectedFrom As String
Local	toolName As String tID As String stringPattern As String dlgResult As DialogResult

New Hire



Form Components

Form	mdiNewHire

Buttons	btnAddCustomer DateTimePicker1 btnRemoveSelectedTool btnAddTool btnCancel btnFinalizeHire
Labels	lblCustomerName lblTotalPrice
Text Boxes	txtCustomerName lstHireToolList txtOrderTotal

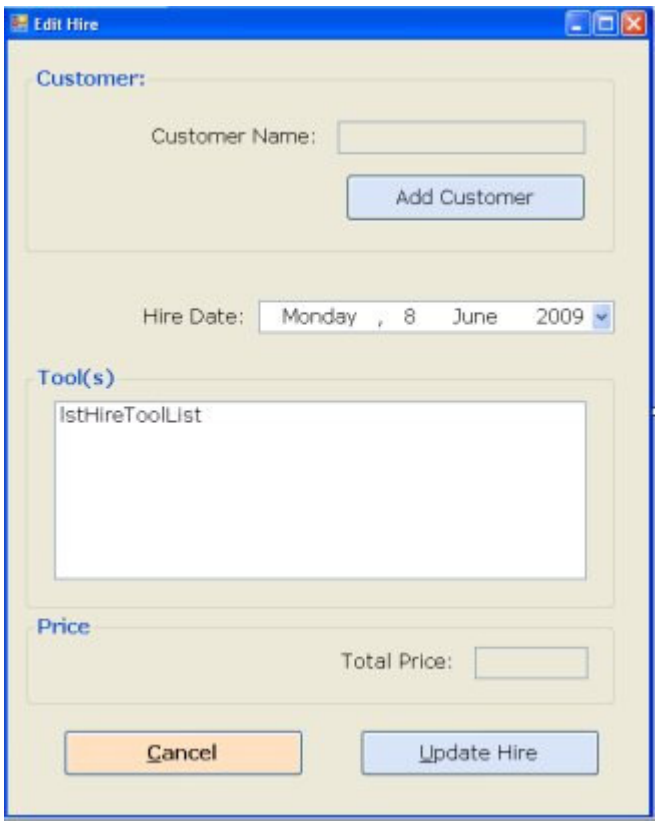
Functional Procedures

Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnAddCustomer	mdiFindCustomer
DateTimePicker1	Library function, return selected date.
btnRemoveSelectedTool	Remove selected tool from lstHireToolList
btnAddTool	mdiFindTool
btnCancel	Cancel update Close form
btnFinalizeHire	Update database

Object variables

Scope	Name
Global	Private Shared anInstance As mdiNewHire Private Business As businessClass Private customerID As String Private toolID As ArrayList
Local	mdiFindCustomer As mdiFindCustomer mdiFindTool As mdiFindTool

Edit Hire



Form Components

Form	mdiEditHire
Buttons	btnAddCustomer DateTimePicker1 btnCancel btnUpdateHire
Labels	lblCustomerName lblHireDate lblTotalPrice
Text Boxes	txtCustomerName lstHireToolList txtOrderTotal

Functional Procedures

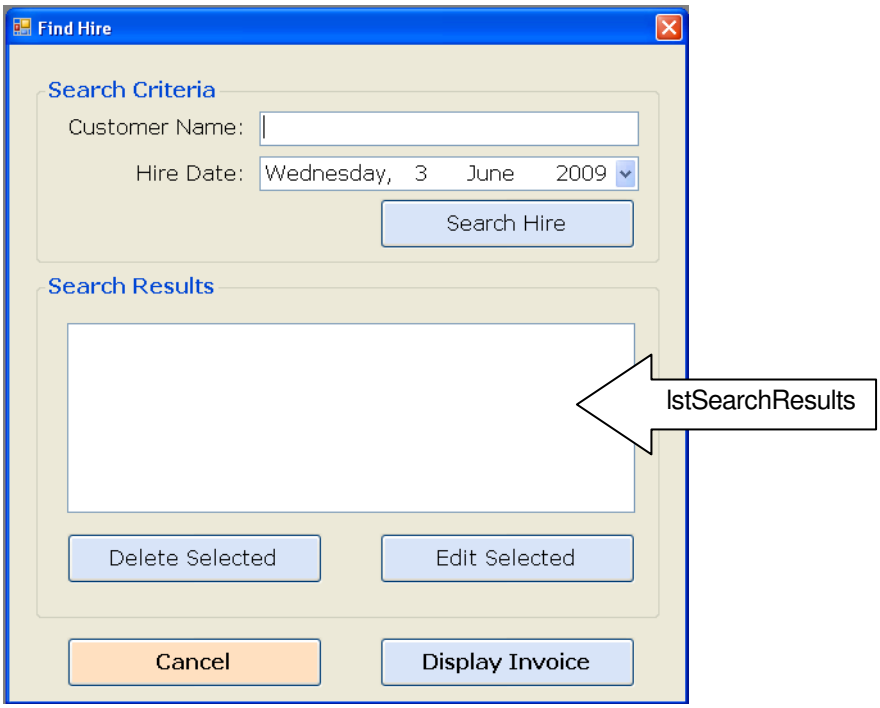
Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
btnAddCustomer	mdiFindCustomer
DateTimePicker1	Library function, return selected date.
btnAddTool	mdiFindTool
btnCancel	Cancel update

	Close form
btnUpdateHire	Update database

Object variables

Scope	Name and Type
Global	Private Shared anInstance As mdiNewHire Private Business As businessClass Private customerID As String Private toolID As ArrayList
Local	mdiFindCustomer As mdiFindCustomer mdiFindTool As mdiFindTool

Find Hire



Form Components

Form	mdiFindHire
Buttons	DateTimePicker1 btnDeleteSelected btnSearchHire btnEditSelected btnCancel btnDispInvoice
Labels	lblCustomerName lblToolName
Text Boxes	txtCustName

	txtToolName lstSearchResults

Functional Procedures

Form_load	Display form and controls Instantiate the data set Instantiate the Binding Source Bind form controls to Binding Source
DateTimePicker1	Library function, return selected date.
btnSearchHire	Search for Hire
btnDeleteSelected	Delete selected Hire from Database
btnEditHire	mdiEditHire
btnCreateCustomer	Update database
btnCancel	Cancel update Close form
btnDispInvoice	DisplayInvoice()

Object variables

Scope	Name and Type
Global	dataClass As dataClass dataClassSet As ToolHireDataSet custBindingSource As BindingSource strCustID As String
Local	msbxAns As DialogResult Dim dateAndTimeDate As Date Dim strMessage As String Dim deleteDialogResult As DialogResult intCustID As Integer

References

Crowley, J. (2001). Data Types [Electronic Version], 1. Retrieved May-2009 from http://www.developerfusion.com/article/32/data-types/.

John W. Satzinger, R. L. J., Stephen D. Burd. (2009). *Systems & Analysis Design in a Changing World* (5th ed.). Boston: Course Technology.

Microsoft. (2003). Microsoft Consulting Services Naming Conventions for Visual Basic [Electronic Version], 1. Retrieved May-2009 from http://support.microsoft.com/kb/110264.