

## **System Maintenance**

### **Overview**

The room hire system has been coded using the language called Visual Basic .NET. My reason for choosing this language in order to create the system was its relative ease of use and the wide range of libraries and functions that it provides. The language is also very well documented and has very good support on the windows platform which I make use of, both at home and in school. As mentioned before, the libraries available for Visual Basic .NET are of great use when creating a project of this scale as they help speed up the development process. An example of this is the Microsoft Word Object Library which allowed me to automate the creation of editable Microsoft Word documents which contain reports concerning bookings created by the system. Without this library it would have been very difficult for me to fulfil the objectives given to me by the user.

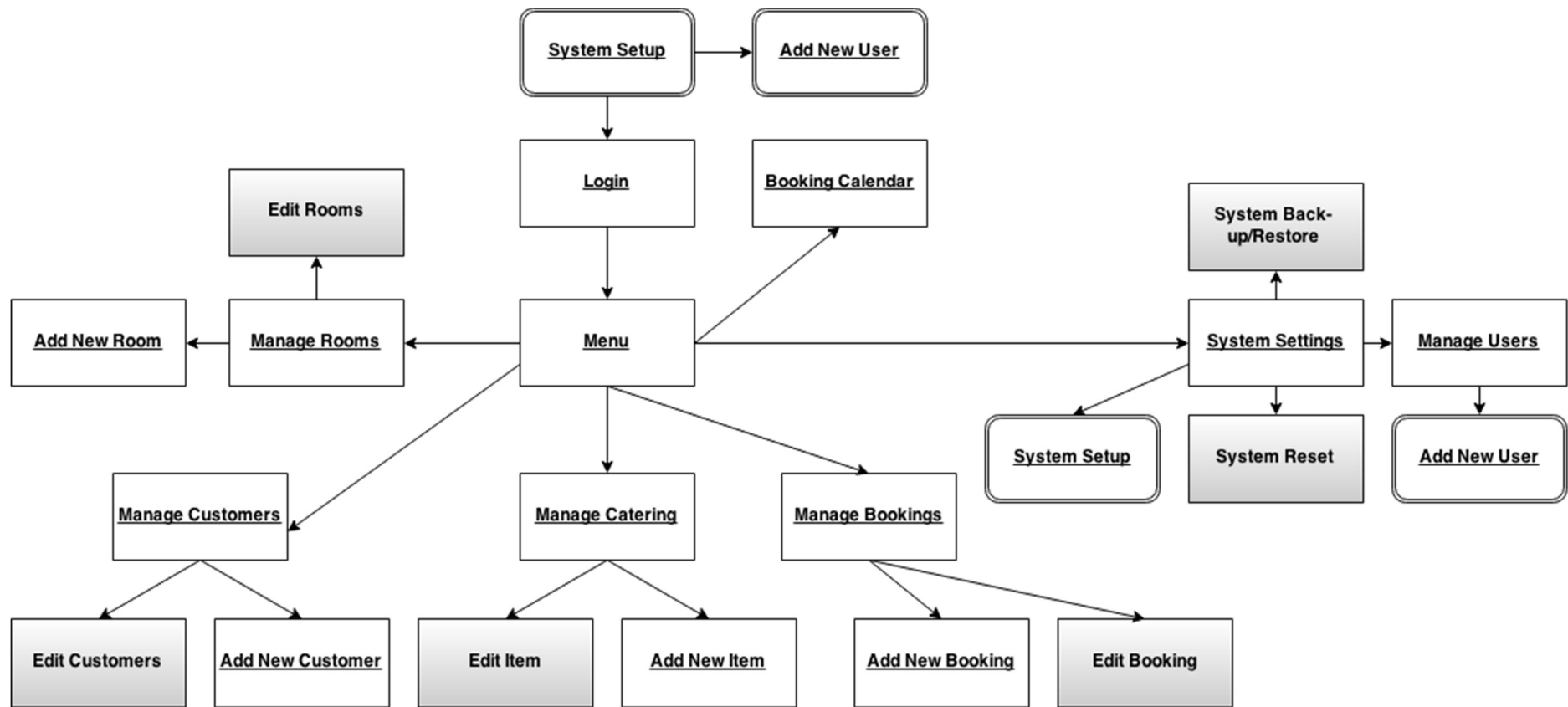
In regards to its documentation, the language Visual Basic. NET was designed by Microsoft and hence various code examples and tutorials can be found on its Microsoft Developer Network website.

The Integrated Development Environment (IDE) I used to develop the system is called Microsoft Visual Studio 2013. My reason for choosing this IDE was the fact that it was available both in school and at home, meaning that there was no need to be switching between different IDE's which could lead to problems during development, for example compatibility issues. Visual Studio provides full support for Visual Basic .NET and was therefore the best choice of an IDE for me to use. It also features auto-correction and an inbuilt interpreter which allows the code to be modified during runtime.

### **Explanation of Modular Structure**

The system is made up of 15 different forms which can be accessed by users and 1 main module which contains global variables, type definitions, common procedures and functions which are often used in the code. Each of the forms may also contain their own variables, type definitions and events which will allow them to function as intended.

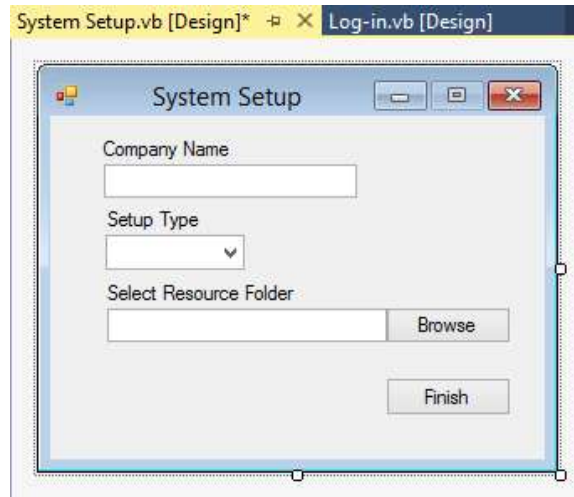
The diagram below shows all of the forms in the system and some of their functions. Any name which is underlined signifies a form while any box with a double outline is a form which can be accessed via more than one route.



## **Form Design View**

### **Setup Form**

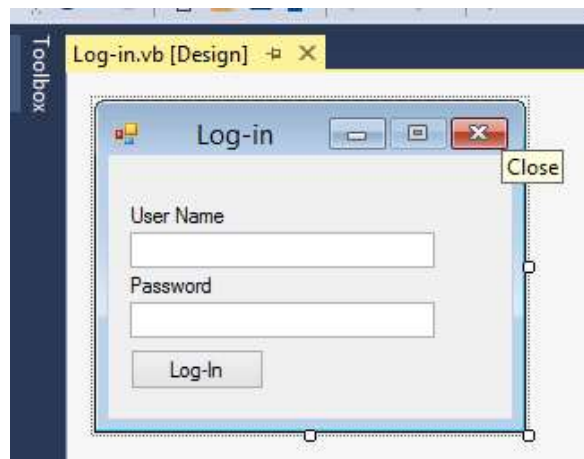
The setup form will be shown on startup if the system or user setup has not been performed. This form will also be shown if the resource folder location contained in the setup file is invalid.



The screenshot shows the design view of the 'System Setup' form. The form has a title bar with 'System Setup' and standard window controls. It contains three main input fields: 'Company Name' (a text box), 'Setup Type' (a dropdown menu), and 'Select Resource Folder' (a text box with a 'Browse' button next to it). A 'Finish' button is located at the bottom right of the form.

### **Login Form**

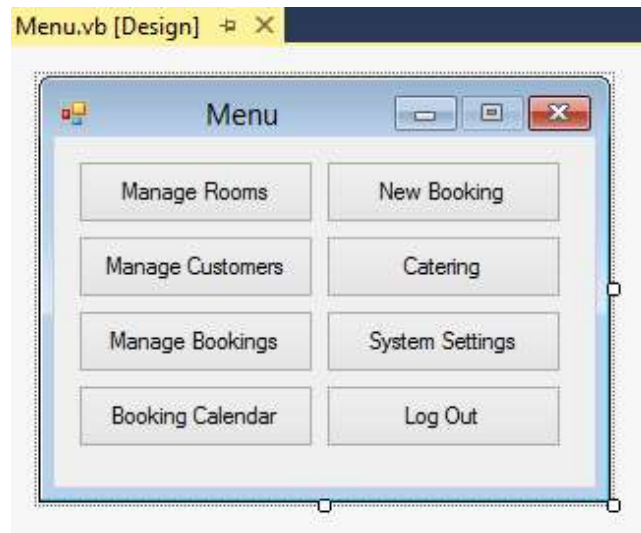
This form will load if the setup has already been set up. The form allows the user to log into the system by entering in their details, namely, their User name and Password. If the user is successful in logging in then the form will be hidden and the Menu form will be shown.



The screenshot shows the design view of the 'Log-in' form. The form has a title bar with 'Log-in' and standard window controls. It contains two input fields: 'User Name' and 'Password' (both text boxes). A 'Log-In' button is located below the password field. A 'Close' button is located at the top right of the form. The form is shown within a design environment with a 'Toolbox' visible on the left.

## Menu Form

The Menu form will be loaded whenever the user log into the system. This form allows the user to navigate the system by pressing the relevant buttons. For example, pressing the Catering button will bring up a form which allows the user to view and edit any catering details.



## Booking Calendar Form

The booking calendar form is loaded whenever the “Booking Calendar” button on the Menu form. This form allows the users to view all booking which take place in a particular time period. The form also allows the user to create a “Booking Report” which contains all details of a selected booking and is presented in the form of a Microsoft Word document.

The image shows the design view of a Windows form titled 'Booking Calendar'. The form has a title bar with standard Windows window controls. The main area of the form is divided into several sections. On the left, there is a table with columns: 'Customer Name', 'Reference', 'Time', 'Room Name', and 'Date'. Below this table, there are input fields for 'Processed by:', 'Room Layout', 'Technical Support', 'Extra Requests', and 'Microphones'. On the right, there is a calendar for 'March 2015' showing days of the week and dates. Below the calendar, there are buttons for 'Month' and 'Year' under the label 'Show bookings for current:'. At the bottom right, there is a table with columns: 'Name', 'Quantity', and 'Delivery Time'. At the bottom center, there are buttons for 'Create Report' and 'Close'. The form is shown within a window titled 'BookingCalendar.vb [Design]'.

## New Booking Form

The New Booking Form will be loaded when the user presses the “New Booking” on the Menu. The form allows the user to create a new booking for an existing customer or for a new customer if necessary.

The screenshot shows a Windows application window titled "New Booking". The window has a standard Windows title bar with minimize, maximize, and close buttons. The main content area is divided into several sections:

- Customer Information:** Includes fields for Name, Telephone Number, Mobile Phone Number, Email Address, Postal Address, and Postcode. There are also fields for Invoicing Contact Name, Invoicing Telephone Number, Invoicing Mobile Phone Number, Invoicing Email Address, Invoicing Postal Address, and Invoicing Postcode.
- Booking Details:** Includes fields for Booking Reference, Booking Date (set to 23 March 2015), Start Time (10:48:59), End Time (10:48:59), and a Select Room dropdown.
- Room Setup:** Includes a Room Setup section with a Layout field, a Tech Support field, and a Number of Microphones spinner (set to 0).
- Customer Selection:** Includes a "Select Customer" section with a text box labeled "IstCustomer" and a "New Customer" checkbox.
- Item Selection:** Includes a "Select Item" section with a text box labeled "IstItem", a "Delivery Time" dropdown (set to 10:48:59), a "Quantity" spinner (set to 0), and buttons for "Remove Selected Item" and "Add Item".
- Extra Requests:** Includes a text box labeled "Extra Requests".

At the bottom of the form, there are "Finish" and "Cancel" buttons.

## Manage Bookings Form

This form is loaded whenever the “Manage Bookings” button on the Menu form is pressed. This form allows the user to edit any existing bookings.

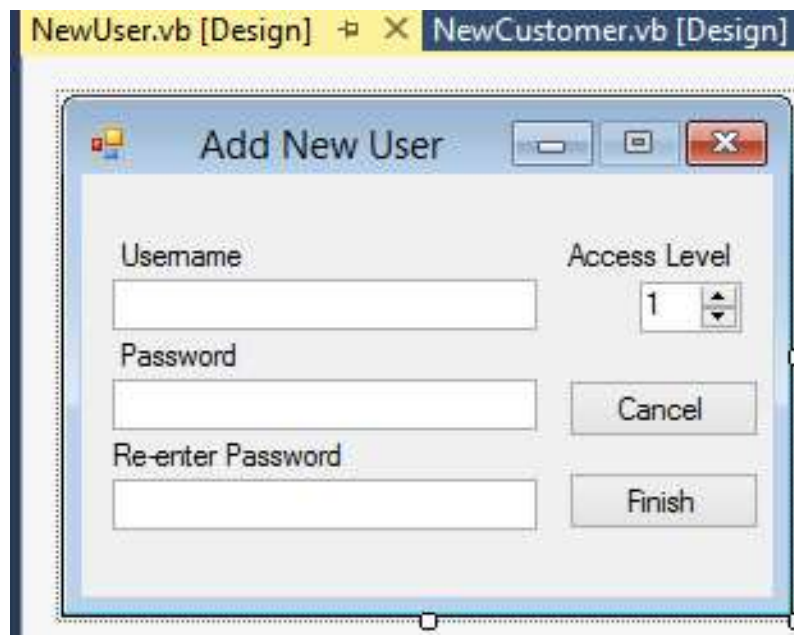
The screenshot shows the 'ManageBookings' form in design view. The form has a title bar 'ManageBookings' with standard window controls. The main area is divided into several sections:

- Select Booking:** A table with columns 'Reference', 'Date', and 'Room'. The table is currently empty.
- Select Customer:** A text box labeled 'IstCustomer'.
- Select Item:** A text box labeled 'IstItem'.
- Delivery Time:** A text box showing '10:50:31' with a calendar icon, a 'Clear Time' button, a 'Remove Selected Item' button, and an 'Add Item' button.
- Quantity:** A text box showing '0' with a spinner control.
- Booking Reference:** A text box.
- Booking Date:** A text box showing '23 March 2015' with a calendar icon.
- Start Time:** A text box showing '10:50:31' with a calendar icon.
- End Time:** A text box showing '10:50:31' with a calendar icon.
- Select Room:** A dropdown menu.
- Layout:** A text box.
- Tech Support:** A text box.
- Status:** A dropdown menu.
- Number of Microphones:** A text box showing '0' with a spinner control.
- Extra Requests:** A text box.

At the bottom of the form are two buttons: 'Close' and 'Update Booking'.

### Add New User Form

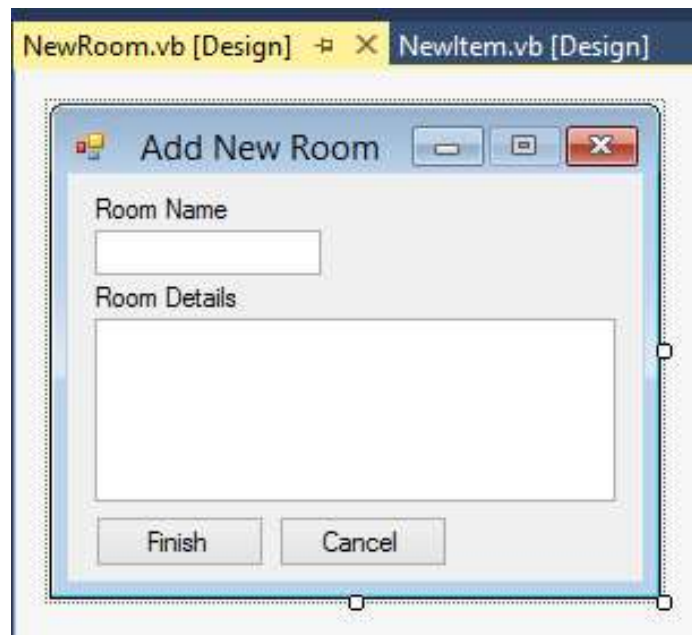
This form is loaded whenever a user presses the “Add New User” button on the Manage Users form. This form allows an administrator to create a new user.



The image shows a screenshot of a Visual Basic IDE with two open windows: 'NewUser.vb [Design]' and 'NewCustomer.vb [Design]'. The 'Add New User' form is displayed in the design view. The form has a title bar with the text 'Add New User' and standard Windows window controls (minimize, maximize, close). The form contains three text input fields: 'Username', 'Password', and 'Re-enter Password'. To the right of the 'Password' and 'Re-enter Password' fields are two buttons: 'Cancel' and 'Finish'. Above the 'Password' field is an 'Access Level' label and a spinner box containing the number '1'. The form is surrounded by a dotted border, indicating it is in design mode.

### Add New Room Form

The form will load when the “Add New Room” button is pressed on the Manage Rooms form. This form allows the user to add a new room



The image shows a screenshot of a Visual Basic IDE with two tabs: 'NewRoom.vb [Design]' and 'NewItem.vb [Design]'. The 'Add New Room' form is displayed in the design view. The form has a title bar with the text 'Add New Room' and standard window controls (minimize, maximize, close). Inside the form, there is a label 'Room Name' above a text box. Below this is a label 'Room Details' above a larger text area. At the bottom of the form, there are two buttons: 'Finish' and 'Cancel'.



### Add New Item Form

This form is loaded whenever the “Add New Item” button is pressed on the “Manage Catering form. The form allows the user to create a new item.



The image shows a screenshot of a Visual Basic form titled "Add New Item" in a design view. The form has a title bar with the text "NewItem.vb [Design]" and standard window controls (minimize, maximize, close). The form itself has a title bar with the text "Add New Item" and standard window controls. It contains two text boxes: "Name" and "Price (£)". Below the "Name" text box is a text input field. Below the "Price (£)" text box is a text input field. At the bottom of the form are two buttons: "Finish" and "Cancel".

## Add New Customer Form

This form is loaded whenever the “Add New Customer” button is pressed on the Manage Customers form is pressed. The form allows the user to create a new Customer.

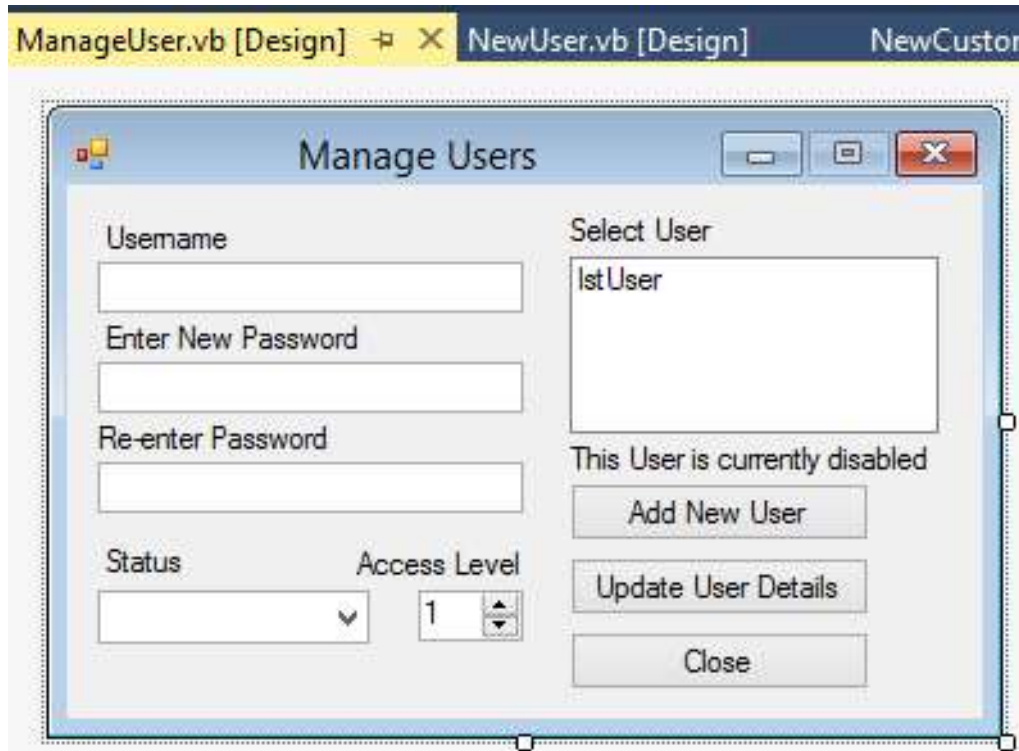
The screenshot shows a Visual Basic IDE with two open files: 'NewCustomer.vb [Design]' and 'NewItem.vb [Design]'. The 'Add New Customer' form is displayed in design view. The form has a title bar with the text 'Add New Customer' and standard Windows window controls (minimize, maximize, close). The form contains a grid of text boxes for data entry, organized into three columns and four rows. The labels for the text boxes are: Name, Telephone Number, Mobile Phone Number, Email Address, Postal Address, Postcode, Invoicing Contact Name, Invoicing Telephone Number, Invoicing Mobile Phone Number, Invoicing Email Address, Invoicing Postal Address, and Invoicing Postcode. At the bottom of the form are two buttons: 'Finish' and 'Cancel'.

Name	Telephone Number	Mobile Phone Number
<input type="text"/>	<input type="text"/>	<input type="text"/>
Email Address	Postal Address	Postcode
<input type="text"/>	<input type="text"/>	<input type="text"/>
Invoicing Contact Name	Invoicing Telephone Number	Invoicing Mobile Phone Number
<input type="text"/>	<input type="text"/>	<input type="text"/>
Invoicing Email Address	Invoicing Postal Address	Invoicing Postcode
<input type="text"/>	<input type="text"/>	<input type="text"/>

Finish Cancel

## Manage Users Form

The Manage Users form will be loaded when the “Manage Users” button is pressed on the System Settings form. This form allows the administrator to edit the details of users.



The screenshot shows a Visual Basic IDE with three open files: `ManageUser.vb [Design]`, `NewUser.vb [Design]`, and `NewCustom`. The `ManageUser.vb [Design]` window is active, displaying a form titled "Manage Users". The form has a standard Windows window border with minimize, maximize, and close buttons. The form's layout includes:

- Username**: A text box for entering the username.
- Enter New Password**: A text box for entering a new password.
- Re-enter Password**: A text box for re-entering the password.
- Status**: A dropdown menu with a downward arrow.
- Access Level**: A numeric spinner box currently set to 1.
- Select User**: A text box containing the text "1st User".
- This User is currently disabled**: A label indicating the user's status.
- Buttons**: Three buttons are located at the bottom right: "Add New User", "Update User Details", and "Close".

## Manage Rooms Form

This form is loaded when the “Manage Rooms” button is pressed on the Menu form.  
The form allows the user to edit room details such as the name of the room.

The image shows a screenshot of a Visual Basic IDE with two open files: `ManageRooms.vb [Design]` and `ManageCustomers.vb [Design]`. The `ManageRooms.vb` form is displayed in the design view. The form has a title bar with the text "Manage Rooms" and standard Windows window controls (minimize, maximize, close). The form's layout includes:

- A "Select Room" section with a text box labeled "lstRoom".
- A message "This room is currently disabled" displayed in red text.
- A "Room Name" label followed by a text box.
- A "Room Details" label followed by a large text box.
- A "Status" label followed by a dropdown menu.
- Three buttons: "Update Room Details", "Close", and "Add New Room".

## Manage Customers Form

This form will be loaded when the “Manage Customers” button is pressed on the Menu form. This form allows the user to edit the customers’ details such as their phone number or address.

ManageCustomers.vb [Design]\* X ManageCatering.vb [Design] ManageBookings.vb [Design]

**Manage Customers**

Select Customer  
lstCustomer

This customer is currently inactive

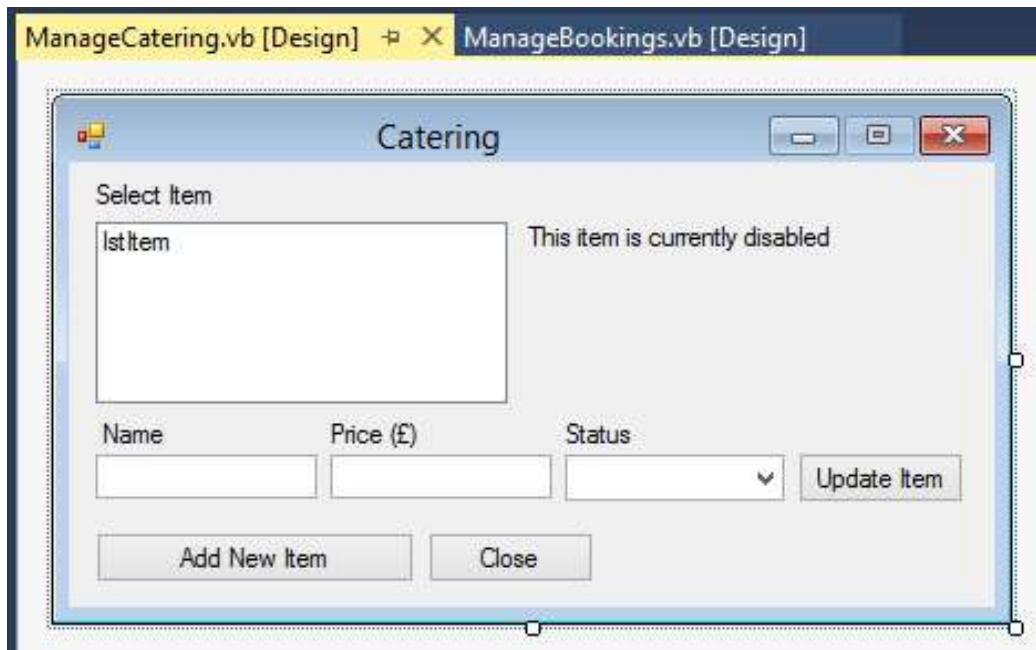
Status  
▼

Update Customer Details Add New Customer Close

Name	Telephone Number	Mobile Phone Number
Email Address	Postal Address	Postcode
Invoicing Contact Name	Invoicing Telephone Number	Invoicing Mobile Phone Number
Invoicing Email Address	Invoicing Postal Address	Invoicing Postcode

## Manage Catering Form

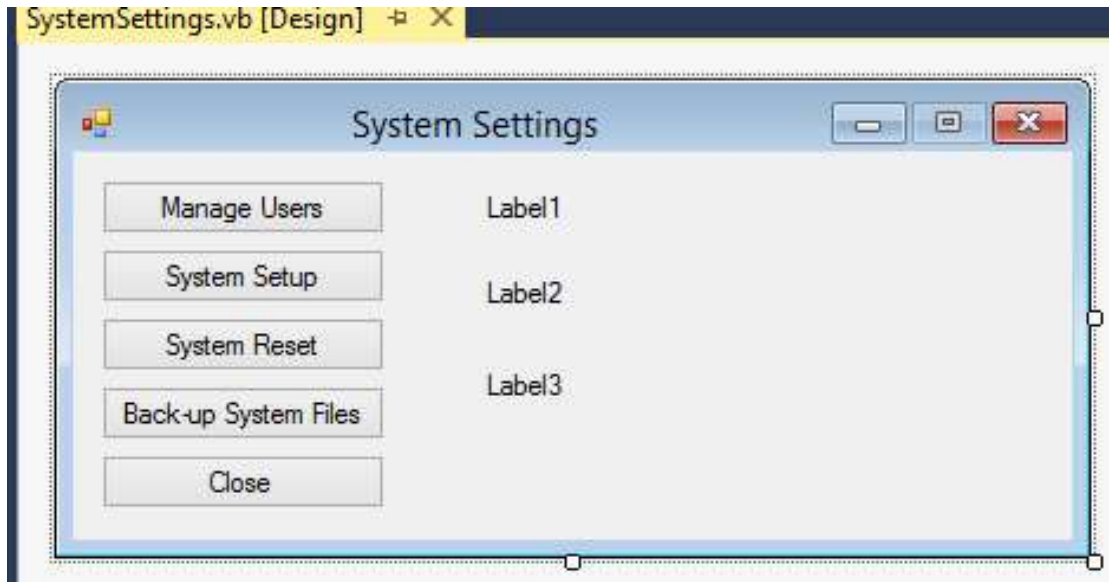
The form will be loaded whenever the “Catering” button is pressed on the Menu form. This form allows the user to change the details of an item such as their price.



The image shows a screenshot of a Visual Studio IDE with two tabs open: 'ManageCatering.vb [Design]' and 'ManageBookings.vb [Design]'. The 'ManageCatering.vb [Design]' tab is active, displaying a Windows form titled 'Catering'. The form has a standard Windows title bar with minimize, maximize, and close buttons. Inside the form, there is a section titled 'Select Item' containing a large text box labeled 'lstItem'. To the right of this text box, the text 'This item is currently disabled' is displayed. Below the 'Select Item' section, there are three input fields: 'Name', 'Price (£)', and 'Status'. The 'Status' field is a dropdown menu. To the right of these fields is an 'Update Item' button. At the bottom of the form, there are two buttons: 'Add New Item' and 'Close'.

## **System Settings Form**

The System Settings form will be loaded when an administrator presses the “Settings” button on the Menu form. This form allows the administrator to change the system settings.



## **Component Acronyms**

Each component name begins with a three letter acronym which can be used to identify the type of component. Below is a list of these acronyms and their descriptions

“txt” – A text box which enables user to enter text

“btn” – A simple button which can be pressed to trigger an event. Can contain text

“lbl” – A label which displays text on the screen

“cmb” – A Combo box which displays an editable text box with a drop down list of allowable values

“dgv” – Data grid view which displays rows and columns of data in a customizable grid

“cbx” – Check box which allows the user to select an associated option

“mtb” – Masked text box allows for the concealment of characters entered in to it

“pnl” – Panel holding a group of related components

## **System variables, procedures and structures**

## Global variables

Name	Type	Description
Access Level	Public Integer	The access level of the current user
UserID	Public Integer	The User ID of the current user

## Structures

Name	Variables	Description
SystemSetupData	CompanyName as String ResourceFolder as String	A type storing system setup information.
UserData	UserID as Integer Name as String Password as String AccessLevel as Integer Active as Boolean	A type storing the data of a system user.
CustomerData	Name as String Telephone as String MobilePhone as String Email as String PostalAddress as String PostCode as String IName as String ITelephone as String IMobilePhone as String IEmail as String IPostalAddress as String IPostCode as String BookingPointer as Integer Active as Boolean	A type storing the data of a customer.
RoomData	RoomID as Integer Name as String Details as String Active as Boolean	A type storing data about a room
CateringData	ItemID as Integer Name as String Price as Integer Active as Boolean	A type storing data about an item
OrderData	ItemID as Integer Quantity as Integer DeliveryTime as Date	A type storing data about a single customer order
ExtraRequestData	Details as String	A type storing data about extra catering requests
RoomSetupData	Layout as String Microphones as Integer TechSupport as String	A type storing data about room setup
BookingData	BookingID as Integer CustomerID as Integer	A type storing data about a particular booking.



	RoomID as Integer UserID as Integer Ref as String BookingDate as Date Start Time a Date End Time as Date CustomerOrderFile as String RoomLayoutFile as String RequestFile as String NextBooking as Integer Active as Boolean	
CustomerControlData	InUse as Boolean Max as Integer Current as Integer	A type holding the control data for the Customer Control File
RoomControlData	InUse as Boolean Max as Integer Current as Integer	A type holding the control data for the Room Control File
CateringControlData	InUse as Boolean Max as Integer Current as Integer	A type holding the control data for the Catering Control File
BookingControlData	InUse as Boolean Max as Integer Current as Integer	A type holding the control data for the Booking Control File
UserControlData	InUse as Boolean Max as Integer Current as Integer	A type holding the control data for the User Control File

## Procedures and Functions

Name	Local variables	Description
Unpad	i as Integer NewLength as Integer NewWord as String	Function which removes empty spaces from the end of strings. Takes a string as a parameter and returns a string with no spaces at the end.
LockCustomerFile		Locks Customer File to prevent other users from accessing it. Takes the resource folder location and the relevant file control as parameters
UnlockCustomerFile		Unlocks the Customer File. Takes the resource folder location and the relevant file control as

		parameters
LockUserFile		Locks User File to prevent other users from accessing it. Takes the resource folder location and the relevant file control as parameters
UnlockUserFile		Unlocks the User File. Takes the resource folder location and the relevant file control as parameters
LockRoomFile		Locks Room File to prevent other users from accessing it. Takes the resource folder location and the relevant file control as parameters
UnlockRoomFile		Unlocks the Room File. Takes the resource folder location and the relevant file control as parameters
LockBookingFile		Locks Booking File to prevent other users from accessing it. Takes the resource folder location and the relevant file control as parameters
UnlockBookingFile		Unlocks the Booking File. Takes the resource folder location and the relevant file control as parameters
LockCateringFile		Locks Catering File to prevent other users from accessing it. Takes the resource folder location and the relevant file control as parameters
UnlockCateringFile		Unlocks the Catering File. Takes the resource folder location and the relevant file control as parameters
CheckForClashes	i as Integer Booking as BookingData NumberOfClashes as Integer ClashingBooking Array [20] of BookingData Clash as Boolean	Function which returns a booking which clashes with the booking described by the parameters passed to it. Takes the parameters: BookingID ,RoomID, BookingDate ,StartTime,

		EndTime , ResourceFolder and BookingControl.
--	--	--

### **Class – frmLogin**

Class representing the Log-in form which allows the users to gain access into the system

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Stores the system setup information
UserControl	UserControlData	Stores control information for the user file
User	UserData	Stores user information

#### **Events**

<b>Name</b>	<b>Local variables</b>	<b>Description</b>
frmLogin_Load	SetupForm as frmSetup	Handles the form load event for the form. Initiates system setup if the setup file does not exist.
btnLogin_Click	MenuForm as frmMenu Found as Boolean i as Integer	Handles a button click. Allows the user to log in if valid credentials have been entered. If the user manages to log in the Menu form will be displayed.

### **Class – frmBookingCalendar**

Class representing the Booking Calendar form

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Order	OrderDetails	Holds the order details concerning the currently selected item
Setup	SystemSetupData	Holds the system setup information
UserControl	UserControlData	Holds control information for the user file
User	UserData	Holds user information
Item	CateringData	Holds information about

		an item which can be ordered
ItemControl	CateringControlData	Holds control information for the catering file
Booking	BookingData	Holds information concerning a booking
BookingControl	BookingControlData	Holds control information for the booking file
Customer	CustomerData	Holds information about a customer
CustomerControl	CustomerControlData	Holds control information for the customer file
Room	RoomData	Holds information about a room
RoomControl	RoomControlData	Holds control information for the room file
RoomSetup	RoomSetupData	Holds room setup information
Request	ExtraResquestData	Holds extra catering request data

## Events

Name	Local variables	Description
calBooking_DateChanged		Date changed event for <b>calBooking</b> . Loads the bookings taking place on the selected day/days into <b>dgvBookingResults</b>
frmBookingCalendar_Load		Form load event. All bookings taking place within the current month are loaded into <b>dgvBookingResults</b>
btnCloseClick		Handles <b>btnClose</b> click event. Closes the form
dgvBookingResults_CellClick	i as Integer y as Integer	Handles a cell click within <b>dgvBookingResults</b> . Loads information about the selected booking into text boxes and <b>dgvOrder</b>
btnCreateReport_Click	WordApp as Word.Application WordDoc as Word.Document WordParagraph as Word.Paragraph WordTable as Word.Table	Handles the <b>btnCreateReport</b> click event. Creates and editable Microsoft Word document and displays it on the screen.

	y as Integer	
btnMonth_Click		Handles the <b>btnMonth</b> click event. Bookings taking place within the currently selected month are loaded into <b>dgvBookinResults</b>
btnYear_Click		Handles the <b>btnYear</b> click event. Bookings taking place within the currently selected year are loaded into <b>dgvBookinResults</b>

### **Class – frmNewBooking**

Class representing the New Booking form. Allows the user to create a brand new booking for a user.

#### **Structures**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
OrderDetails	ItemID as Integer Name as String Quantity as Integer DeliveryTime as Date	Structure storing information about a single item ordered by a customer.

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Order	Array[200] of OrderDetails	An array storing each item ordered by a customer
NoOfItems	Integer	Holds the number of items ordered by a customer
Setup	SystemSetupData	Holds the system setup information
UserControl	UserControlData	Holds control information for the user file
User	UserData	Holds user information
Item	CateringData	Holds information about an item which can be ordered
ItemControl	CateringControlData	Holds control information for the catering file
Booking	BookingData	Holds information concerning a booking
BookingControl	BookingControlData	Holds control information

		for the booking file
Customer	CustomerData	Holds information about a customer
CustomerControl	CustomerControlData	Holds control information for the customer file
Room	RoomData	Holds information about a room
RoomControl	RoomControlData	Holds control information for the room file
RoomSetup	RoomSetupData	Holds room setup information
Request	ExtraResquestData	Holds extra catering request data

## Events

Name	Variables	Description
frmNewBooking_Load		Handles the form load event. Populates <b>lstCustomer</b> , <b>lstItem</b> and <b>cmbRoom</b>
btnAddBooking_Click	i as Integer LastBooking as Integer NewOrder as OrderData ClashingBooking as BookingData	Handles the <b>btnAddBooking</b> click event. Uses the information provided by the customer to create a new booking and add it to the booking file
cbxNewCustomer_CheckedChanged		Handles the check state changed event for <b>cbxNewCustomer</b> . Hides or displays the panel ( <b>pnlNewCustomer</b> ) containing text boxes used for inputting new customer information.
btnAddItem_Click		Handles the <b>btnAddItem</b> click event. Adds a new item to <b>dgvItems</b>
btnRemoveSelected_Click	CurrentIndex as Integer	Handles the <b>btnRemoveSelected</b> click event. Removes the selected item from <b>dgvItems</b> .
btnClearTime_Click		Handles the <b>btnClearTime</b> click event. Resets the delivery time for an item.
lstCustomer_SelectedIndex		Handles the <b>lstCustomer</b>

x Changed		selected index changed event. Checks if the selected customer is active. If the customer is inactive then an error message will be shown and the user will be prevented from choosing the customer.
cmbRoom_SelectedIndex Changed		Handles the <b>cmbRoom</b> selected index changed event. Checks whether the currently selected room is active.
lstItem_SelectedIndex Changed		Handles the <b>lstItem</b> selected index changed event. Checks whether the currently selected item is active.
btnCancel_Click		Handles the <b>btnCancel</b> click event. Closes the form.
dtpStartTime_Value Changed		Handles the <b>dtpStartTime</b> value changed event. Changes the value of <b>dtpDelivery</b> to the start time of the booking.

### Class – frmManageBookings

Class representing the Manage Bookings form. Allows a user to modify existing bookings.

### Structures

Name	Variables	Description
OrderDetails	ItemID as Integer Name as String Quantity as Integer DeliveryTime as Date	Structure storing information about a single item ordered by a customer.

### Variables

Name	Type	Description
Order	Array[200] of OrderDetails	An array storing each item ordered by a customer

NoOfItems	Integer	Holds the number of items ordered by a customer
Setup	SystemSetupData	Holds the system setup information
UserControl	UserControlData	Holds control information for the user file
User	UserData	Holds user information
Item	CateringData	Holds information about an item which can be ordered
ItemControl	CateringControlData	Holds control information for the catering file
Booking	BookingData	Holds information concerning a booking
BookingControl	BookingControlData	Holds control information for the booking file
Customer	CustomerData	Holds information about a customer
CustomerControl	CustomerControlData	Holds control information for the customer file
Room	RoomData	Holds information about a room
RoomControl	RoomControlData	Holds control information for the room file
RoomSetup	RoomSetupData	Holds room setup information
Request	ExtraResquestData	Holds extra catering request data

## Events

Name	Variables	Description
frmNewBooking_Load		Handles the form load event. Populates <b>lstCustomer</b> , <b>lstItem</b> and <b>cmbRoom</b>
btnUpdate_Click	NewOrder as OrderData ClashingBooking as BookingData y as Integer	Handles the <b>btnUpdate</b> click event. Updates the booking information held in system files.
dgvBookings_CellClick	NewOrder as OrderData y as Integer	Handles the dgvBookings cell click event. Retrieves booking information about the currently selected booking.
btnAddItem_Click		Handles the <b>btnAddItem</b> click event. Adds a new item to <b>dgvItems</b>



btnRemoveSelected_Click	CurrentIndex as Integer	Handles the <b>btnRemoveSelected</b> click event. Removes the selected item from <b>dgvItems</b> .
btnClearTime_Click		Handles the <b>btnClearTime</b> click event. Resets the delivery time for an item.
lstCustomer_SelectedIndex Changed		Handles the <b>lstCustomer</b> selected index changed event. Checks if the selected customer is active. If the customer is inactive then an error message will be shown and the user will be prevented from choosing the customer.
cmbRoom_SelectedIndex Changed		Handles the <b>cmbRoom</b> selected index changed event. Checks whether the currently selected room is active.
lstItem_SelectedIndex Changed		Handles the <b>lstItem</b> selected index changed event. Checks whether the currently selected item is active.
btnFinish_Click		Handles the <b>btnCancel</b> click event. Closes the form.
dtpStartTime_Value Changed		Handles the <b>dtpStartTime</b> value changed event. Changes the value of <b>dtpDelivery</b> to the start time of the booking.

### **Class – frmManageCatering**

Class representing the Manage Catering form which allows a user to update the information of each item which can be ordered by a customer.

### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
Item	CateringData	Holds information about

		an item which can be ordered
ItemControl	CateringControlData	Holds control information for the catering file

## Events

Name	Variables	Description
frmManageCatering_Load		Handles the form load event. Populates <b>IstItem</b>
btnClose_Click		Handles the <b>btnClose</b> click event. Closes the form.
IstItem_selectedIndex Changed		Handles the <b>IstItem</b> selected index changed event. Checks if the selected item is active.
btnUpdate_Click		Handles the <b>btnUpdate</b> click event. Updates the item information within the system files.
btnNewItem_Click	AddNewItem as frmNewItem	Handles the <b>btnNewItem</b> click event. Displays the New Item form ( <b>frmNewItem</b> ).

## Class – frmManageCustomers

Class representing the Manage Customers form which allows a user to update the information of each customer.

## Variables

Name	Type	Description
Setup	SystemSetupData	Holds the system setup information
Customer	CustomerData	Holds information about a customer
CustomerControl	CustomerControlData	Holds control information for the customer file

## Events

Name	Variables	Description
frmManageCustomer_Load		Handles the form load event. Populates <b>IstCustomer</b>

btnClose_Click		Handles the <b>btnClose</b> click event. Closes the form.
IstCustomer_selectedIndexChanged		Handles the <b>IstCustomer</b> selected index changed event. Checks if the selected item is active.
btnUpdate_Click		Handles the <b>btnUpdate</b> click event. Updates the customer information within the system files.
btnAddNewCustomer_Click	AddNewCustomer as frmNewCustomer	Handles the <b>btnNewItem</b> click event. Displays the New Customer form ( <b>frmNewCustomer</b> ).

### **Class – frmManageRooms**

Class representing the Manage Rooms form which allows a user to update the information of each room.

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
Room	RoomData	Holds information about a room
RoomControl	RoomControlData	Holds control information for the room file

#### **Events**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
frmManageRoom_Load		Handles the form load event. Populates <b>IstRoom</b>
btnClose_Click		Handles the <b>btnClose</b> click event. Closes the form.
IstRoom_selectedIndexChanged		Handles the <b>IstRoom</b> selected index changed event. Checks if the selected room is active.
btnUpdate_Click		Handles the <b>btnUpdate</b> click event. Updates the room information within the system files.
btnAddNewRoom_Click	AddNewRoom as frmNewRoom	Handles the <b>btnNewRoom</b> click event.

		Displays the New Customer form ( <b>frmNewRoom</b> ).
--	--	---

### **Class – frmManageUsers**

Class representing the Manage Users form which allows an administrator to

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
User	UserData	Holds information about a User
UserControl	UserControlData	Holds control information for the User file

#### **Events**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
frmManageUsers_Load		Handles the form load event. Populates <b>IstUser</b>
btnClose_Click		Handles the <b>btnClose</b> click event. Closes the form.
IstRoom_selectedIndex Changed		Handles the <b>IstUser</b> selected index changed event. Checks if the selected user is active.
btnUpdate_Click		Handles the <b>btnUpdate</b> click event. Updates the user information within the system files.
btnAddNewUser_Click	AddNewUser as frmNewUser	Handles the <b>btnAddNewUser</b> click event. Displays the New Customer form ( <b>frmNewUser</b> ).

### **Class – frmMenu**

Class representing the Menu form which allows the user to navigate the system.

#### **Events**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
btnNewBooking_Click	NewBookingForm as frmNewBooking	Handles <b>btnNewBooking</b> click event. Shows the New Booking form.

btnManageRooms_Click	ManageRoomsForm as frmManageRooms	Handles <b>btnManageRooms</b> click event. Shows the Manage Rooms form.
btnManageCatering_Click	ManageCateringForm as frmManageCatering	Handles <b>btnManageCatering</b> click event. Shows the Manage Catering form.
btnLog_out_Click		Handles <b>btnLog_out</b> click event. Closes the form.
btnManageBookings_Click	ManageBookingsForm as frmManageBookings	Handles <b>btnManageBookings</b> click event. Shows the Manage Bookings form.
btnBookingCalendar_Click	BookingCalendarForm as frmBookingCalendar	Handles <b>btnBookingCalendar</b> click event. Shows the Booking Calendar form.
CloseMenu_Click		Handles the form closed event. Shows the log-in form
btnSystemSettings_Click	SystemSettingsForm as frmSystemSettings	Handles the <b>btnSystemSettings</b> click event. Shows the System Settings form.
frmMenu_Load		Handles the form load event. Disables buttons based on the user's access level.

### **Class – frmNewCustomer**

Class representing the New Customer form which allows the user to add a new customer

### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
Customer	CustomerData	Holds information about a customer
CustomerControl	CustomerControlData	Holds control information for the Customer file

### **Events**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
frmNewCustomer_Load		Handles form load event. Checks if the file needs to be extended.

btnFinish_Click		Handles the <b>btnFinish</b> lick event. Adds new customer to the file.
btnCancel_Click		Handles the <b>btnCancel</b> click event. Closes the form.

### **Class – frmNewItem**

Class representing the New Item form which allows the user to add a new item

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
Item	CateringData	Holds information about an item
ItemControl	CateringControlData	Holds control information for the Catering file

#### **Events**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
frmNewItem_Load		Handles form load event. Checks if the file needs to be extended.
btnFinish_Click		Handles the <b>btnFinish</b> lick event. Adds new item to the file.
btnCancel_Click		Handles the <b>btnCancel</b> click event. Closes the form.

### **Class – frmNewRoom**

Class representing the New Room form which allows the user to add a new Room

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
Room	Room Data	Holds information about a Room
Room Control	Room ControlData	Holds control information for the Room file

#### **Events**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
frmNewRoom_Load		Handles form load event. Checks if the file needs to be extended.
btnFinish_Click		Handles the <b>btnFinish</b> click event. Adds new room to the file.
btnCancel_Click		Handles the <b>btnCancel</b> click event. Closes the form.

### **Class – frmNewUser**

Class representing the New Customer form which allows an administrator to add new user.

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
User	User Data	Holds information about a user
User Control	User ControlData	Holds control information for the User file

#### **Events**

<b>Name</b>	<b>Variables</b>	<b>Description</b>
frmNewUser_Load		Handles form load event. Checks if the file needs to be extended.
btnFinish_Click		Handles the <b>btnFinish</b> click event. Adds new User to the file.
btnCancel_Click		Handles the <b>btnCancel</b> click event. Closes the form.

### **Class – frmSystemSetup**

Class representing the System Setup form which allows a user to perform a system or user setup.

#### **Variables**

<b>Name</b>	<b>Type</b>	<b>Description</b>
Setup	SystemSetupData	Holds the system setup information
UserControl	UserControlData	Holds control information

		for the user file
User	UserData	Holds user information
Item	CateringData	Holds information about an item which can be ordered
ItemControl	CateringControlData	Holds control information for the catering file
Booking	BookingData	Holds information concerning a booking
BookingControl	BookingControlData	Holds control information for the booking file
Customer	CustomerData	Holds information about a customer
CustomerControl	CustomerControlData	Holds control information for the customer file
Room	RoomData	Holds information about a room
RoomControl	RoomControlData	Holds control information for the room file

## Events

Name	Variables	Description
frmSystemSetup_Load		Handles form load event. Checks if the setup file exists and is not blank.
btnFinish_Click		Handles the <b>btnFinish</b> click event. Links the setup file to the resource folder if the user setup is chosen. If the system setup is chosen then the system files will be generated in the specified location.
btnBrowse_Click		Handles the <b>btnBrowse</b> click event. Displays the folder browser dialog ( <b>fbdResource</b> ).

## Class – frmSystemSettings

Class representing the System Settings form which allows an administrator to change system settings.

### Variables

Name	Type	Description
Setup	SystemSetupData	Holds the system setup



		information
Booking	BookingData	Holds information concerning a booking
BookingControl	BookingControlData	Holds control information for the booking file
Customer	CustomerData	Holds information about a customer
CustomerControl	CustomerControlData	Holds control information for the customer file
Room	RoomData	Holds information about a room
RoomControl	RoomControlData	Holds control information for the room file

## Events

Name	Variables	Description
btnManageUsers_Click	ManageUsersForm as frmManageUsers	Handles the <b>btnManageUsers</b> click event. Displays the Manage Users form.
btnSetup_Click	SetupForm as frmSetup	Handles the <b>btnSetup</b> click event. Displays the System Setup form.
btnReset_Click	Result as Integer SetupForm as frmSetup	Handles the <b>btnReset</b> click event. Allows the user to carry out a system reset which will delete all of the system data.
frmSystemSettings_Load	CustomerID as Integer NoOfBookings as Integer RoomID as Integer HighestNoB as Integer HighestNoRU as Integer i as Integer CustomerArray as Variable Length Array of Integer RoomArray as Variable Length Array of Integer	Handles the form load event. Calculates the system statistics.
btnBackUp_Click	Filepath as String	Handles the <b>btnBackUp</b> click event. Creates a back-up of the system and saves it in a location chosen by the user.
btnClose_Click		Handles the <b>btnClose</b> click event. Closes the form.

## Overview of complex algorithms

### Log-In algorithm

The log in algorithm checks that the Username and password contained in the text fields belong to a genuine user in the system. Each Username and password pair contained in the User file is compared to the information entered by the user. If a match is found then the user will be allowed to log in, otherwise an error message will be shown.

#### *Open User File*

*While User Not found And i <= Total number of users Do*

*Retrieve each user from the file from at position i in the user file*

*Compare the username and password with input data*

*If username <> input\_username or Password <> input\_password then*

*User not found*

*Else if the username and password do not match the input data*

*User found*

*Set Access level = User.AccessLevel*

*Set UserID = User.UserID*

*End if*

*i = i+1*

*End While*

*Close User File*

*If User found then*

*Set reset text box values to default*

*Hide Login Form*

*Display Menu form*

*Else if User not found*

*Display Error Message telling the User to try again*

*End if*

```
LockUserFile(Setup.ResourceFolder,UserControl)
FileOpen(1, Setup.ResourceFolder + "\User.dat", OpenMode.Random, , , Len(User))
'open user file
While Found = False And i <= UserControl.Current
    'while user not found And i <= Total number of users
    FileGet(1, User, i)
'Retrieve each user from the file from at position i in the user file
    If Unpad(User.Name) <> txtUserName.Text Or Unpad(User.Password) <>
mtbPassword.Text Then
        Found = False 'User not found
```

```

        ElseIf User.Active = True Then
'else if the username and password do not match the input data
            Found = True 'User found
            AccessLevel = User.AccessLevel
'Set Access level = User.AccessLevel
            UserID = User.UserID 'Set UserID = User.UserID
        End If
        i = i + 1
    End While
    FileClose(1) 'close User file
    UnlockUserFile(Setup.ResourceFolder, UserControl)
    If Found = True Then 'if a match has been found
        'Set reset text box values to default
        txtUserName.Text = ""
        mtbPassword.Text = ""
        Me.Hide() 'hide the log-in form
        MenuForm.ShowDialog() ' show the Menu form
    Else 'if user not found
        'Display Error Message telling the User to try again
        MessageBox.Show("Invalid Log-in credentials. Please try again", "Login
failed", MessageBoxButtons.OK, MessageBoxIcon.Error)
    End If

```

## Linking Bookings

Each booking in the Booking file contains a link to an earlier booking made by the same customer. This algorithm handles the process of linking a new booking for a particular customer with older existing bookings. If the booking currently being processed is the first booking the customer has had then the Booking Pointer field in the customer record will be set to the position of the current booking in the Booking file.

*If Oldest Booking Pointer = 0 then*

*Set Customer.BookingPointer = Number of bookings +1*

*Else Oldest Booking Pointer <> 0 then*

*Open Booking File*

*i = Oldest Booking Pointer*

*While i<>0 do*

*Retrieve Booking from file at position i*

*LastBookingPointer = i*

*i = NextBookingPointer*

*End While*

*Set NextBookingPointer = Number of bookings +1*

*Update booking file at position LastBookingPointer with new  
NextBookingPointer*

*End If*

```

If Customer.BookingPointer = 0 Then 'If Oldest Booking Pointer = 0 then
    Customer.BookingPointer = BookingControl.Current + 1
'Set Customer.BookingPointer = Number of bookings +1
    FileOpen(1, Setup.ResourceFolder + "\Customer.dat",
OpenMode.Random, , , Len(Customer))
    FilePut(1, Customer, 1stCustomer.SelectedIndex + 1)
    FileClose(1)
    UnlockCustomerFile(Setup.ResourceFolder, CustomerControl)
ElseIf Customer.BookingPointer <> 0 Then 'Else Oldest Booking
Pointer <> 0 then
    i = Customer.BookingPointer 'i = Oldest Booking Pointer
    FileOpen(1, Setup.ResourceFolder + "\Booking.dat",
OpenMode.Random, , , Len(Booking)) ' Open Booking File
    While i <> 0
        FileGet(1, Booking, i)
'Retrieve Booking from file at position i
        LastBooking = i 'LastBookingPointer = i
        i = Booking.NextBooking 'i = NextBookingPointer
    End While
    Booking.NextBooking = BookingControl.Current + 1 'Set
NextBookingPointer = Number of bookings +1
    FilePut(1, Booking, LastBooking)
'Update booking file at position LastBookingPointer with new
NextBookingPointer
    FileClose(1)
    UnlockCustomerFile(Setup.ResourceFolder, CustomerControl)
End If

```

## Lock File Procedure

This procedure checks if a file has been opened by a different user. If the file is in use then the program will wait a short amount of time before checking again. If the file is not in use then the InUse field within the particular file will be changed to "TRUE" meaning that no other user will be able to make use of the file. There exists a Lock File procedure for every single file which has a control file. The algorithm below shows the code for the procedure which locks the Room file.

FileControl

*Public Sub Lock\_File(ByVal ResourceFolder As String, ByRef Control As ControlData)*

*Repeat*

*Sleep for 10 milliseconds*

*Open Control file*

*Retrieve Control data*

*Set File Control = Control data*

*Close File*

*Until FileControl.InUse = False*

*Set FileControl.InUse = True*

*Open Control File*

*Update File with FileControl*

*Close File*

```
Public Sub LockRoomFile(ByVal ResourceFolder As String, ByRef Control As
RoomControlData)
    Do 'check if the file is being used by a different user
        Threading.Thread.Sleep(10) 'Sleep for 10 milliseconds
        FileOpen(3, ResourceFolder + "\RoomControl.dat", OpenMode.Random, , ,
Len(Control)) 'Open control file
        FileGet(3, Control, 1) 'Retrieve control data
        FileClose(3) ' close the control file
    Loop Until Control.InUse = False
    Control.InUse = True
    FileOpen(3, ResourceFolder + "\RoomControl.dat", OpenMode.Random, , ,
Len(Control)) 'Open control file
    FilePut(3, Control, 1) 'open control file and update the InUse field
    FileClose(3) ' close the control file
End Sub
```

### Unlock File Procdeure

The Unlock file procedure is used to unlock files so that they may be used by other users. The procedure updates the InUse field in the control file so that it is set to "FALSE". This procedure exists for every file which has a control file.

*Public Sub Unlock\_File(ByVal ResourceFolder As String, ByRef Control As ControlData)*

*Set FileControl.InUse = False*

*Open Control File*

*Update File with FileControl*

*Close File*

```
Public Sub UnlockRoomFile(ByVal ResourceFolder As String, ByRef Control As
RoomControlData)
    Control.InUse = False
    FileOpen(3, ResourceFolder + "\RoomControl.dat", OpenMode.Random, , ,
Len(Control)) 'Open control file
    FilePut(3, Control, 1) 'open control file and update the InUse field
    FileClose(3) ' close the control file
End Sub
```

### Check For Clashes Function

This function returns the information about a booking which has clashes with the booking which was to it in the parameters. This function checks if a booking clashes with an existing booking which is held in the booking file. The algorithm starts by compiling an array of bookings which occur on the same date, in the same room and

are active. Then it checks the booking lies within the time range of all the bookings contained in the list. If a match is found then it will return the clashing booking otherwise an empty booking record will be returned.

```
Public Function CheckForClashes(ByVal BookingID As Integer, ByVal RoomID As Integer,  
ByVal BookingDate As Date,  
ByVal StartTime As Date, ByVal EndTime As Date,  
ByVal ResourceFolder As String,  
ByVal BookingControl As BookingControlData) As BookingData  
Open Booking File
```

```
For i = 1 to NumberOfBookings
```

```
    Retrieve Booking Information from position i in the file
```

```
        If the Room ID =Current Room ID and the Booking date =  
        Current Booking Date then
```

```
            NumberOf Clashes = NumberOf Clashes +1
```

```
            ClashingBooking(NumberOfClashes -1) = Booking  
            Information
```

```
        End If
```

```
End For
```

```
Close booking file
```

```
i = 0
```

```
While( Number Of clashes) <>0 and (Clash <>True) and (i<>Number of Clashes) do
```

```
    If the time period for ClashingBooking(i) lies within that of the Current Booking  
    then
```

```
        Clash = True
```

```
    End If
```

```
i=i+1
```

```
End While
```

```
Booking = Nothing
```

```
If Clash = True then
```

```
    Return Clashing Booking(i-1)
```

```
Else Return Booking
```

```
Public Function CheckForClashes(ByVal BookingID As Integer, ByVal RoomID As Integer,  
ByVal BookingDate As Date, ByVal StartTime As Date, ByVal EndTime As Date,
```

```

ByVal ResourceFolder As String, ByVal
BookingControl As BookingControlData) As BookingData
    Dim i As Integer
    Dim Booking As BookingData = Nothing
    Dim NumberOfClashes As Integer = 0
    Dim ClashingBooking(20) As BookingData
    Dim Clash As Boolean = False
    LockBookingFile(ResourceFolder, BookingControl)
    'Retrieve booking information
    FileOpen(1, ResourceFolder + "\Booking.dat", OpenMode.Random, , ,
Len(Booking))
    For i = 1 To BookingControl.Current 'For i = 1 to NumberOfBookings
        FileGet(1, Booking, i) 'Retrieve Booking Information from position i in
the file
        'check for bookings taking place on the same day and in the same room
        If Booking.RoomID = RoomID And Booking.BookingDate.ToShortDateString =
BookingDate.ToShortDateString And Booking.Active = True And BookingID <>
Booking.BookingID Then
            'If the Room ID =Current Room ID and the Booking date = Current
Booking Date then
                NumberOfClashes = NumberOfClashes + 1
                ClashingBooking(NumberOfClashes - 1) = Booking
            End If
        Next
        FileClose(1) ' close the Booking file
        UnlockBookingFile(ResourceFolder, BookingControl)
        i = 0
        'Check if any of the times clash with each other
        Do While NumberOfClashes <> 0 And (i <> NumberOfClashes) And Clash <> True
            'While( Number Of clashes) <>0 and (Clash <>True) and (i<>Number of
Clashes) do
                If (StartTime.TimeOfDay >= ClashingBooking(i).StartTime.TimeOfDay And
StartTime.TimeOfDay < ClashingBooking(i).EndTime.TimeOfDay) Or
                (EndTime.TimeOfDay > ClashingBooking(i).StartTime.TimeOfDay And
EndTime.TimeOfDay <= ClashingBooking(i).EndTime.TimeOfDay) Or
                (StartTime.TimeOfDay <= ClashingBooking(i).StartTime.TimeOfDay And
EndTime.TimeOfDay >= ClashingBooking(i).EndTime.TimeOfDay) Then
                    'If the time period for ClashingBooking(i) lies within that of the
Current Booking
                        Clash = True
                    End If
                    i = i + 1
                Loop
                Booking = Nothing
                Booking.BookingID = 0
                If Clash = True Then
                    Return ClashingBooking(i - 1)
                Else
                    Return Booking 'return empty booking
                End If
            End Function

```

## Date Changed Event

This event is triggered whenever a user changes the selected date in “calBooking”. The event will load bookings into the dgvBookingResults provided that they take place on the selected date.

### *Sub Date\_Changed Event*

*Clear Information in the data grid*

*For i = 1 to Number of Bookings do*

*Retrieve Booking at position i in the file*

*If the Booking date is within the selected date range then*

*Open Room File*

*Get Room Name for the Room specified in the booking*

*Close Room File*

*Open Customer File*

*Get Customer name for the specified booking from customer file*

*Add the Customer Name, Booking reference, Start time, End Time and  
End Time to data grid Close Customer File*

*End if*

*End For*

*End Sub*

```
Private Sub calBooking_DateChanged(sender As Object, e As DateRangeEventArgs) Handles  
calBooking.DateChanged
```

```
    dgvBookingResults.Rows.Clear() 'Clear Information in the data grid  
    LockBookingFile(Setup.ResourceFolder, BookingControl)  
    FileOpen(1, Setup.ResourceFolder + "\Booking.dat", OpenMode.Random, , ,  
Len(Booking))  
    'Open booking file  
    For i As Integer = 1 To BookingControl.Current  
        FileGet(1, Booking, i)  
  
        If Booking.BookingDate >= calBooking.SelectionStart And  
Booking.BookingDate <= calBooking.SelectionEnd Then  
            FileOpen(2, Setup.ResourceFolder + "\Room.dat", OpenMode.Random, , ,  
Len(Room))  
            'Open room file  
            FileGet(2, Room, Booking.RoomID) 'Get Room Name for the Room specified  
in the booking  
            FileClose(2) 'Close room file  
            FileOpen(2, Setup.ResourceFolder + "\Customer.dat", OpenMode.Random, ,  
, Len(Customer))  
            'Open customer file  
            FileGet(2, Customer, Booking.CustomerID)  
            FileClose(2) 'close customer file  
            dgvBookingResults.Rows.Add(Unpad(Customer.Name), Unpad(Booking.Ref),  
Booking.StartTime.ToShortTimeString & "-" & Booking.EndTime.ToShortTimeString,  
Unpad(Room.Name), Booking.BookingID, Booking.BookingDate.ToShortDateString)  
            'Add the Customer Name, Booking reference, Start time, End Time and  
End Time to data grid Close Customer File  
            End If  
        Next  
        FileClose(1) 'Close booking file
```



```
UnlockBookingFile(Setup.ResourceFolder, BookingControl)
```

```
End Sub
```