AS Computing CG2 Extended Task

SADS Booking System

2012/2013
King Edward VI Aston School, 20147
08LeeM

Contents

Analysis and Design

Product Definition

Background Summary

> ***"Starshine Amateur Dramatic Society (SADS) was established in 1996. The society usually performs two shows each year, a family pantomime in the winter and a play in the summer. The summer plays have ranged from comedies through costume dramas to modern day thrillers."***

The Starshine Amateur Dramatic Society (SADS) requires a computerized system which books and records customer information.

Aims

The software should be able to:

- Record and store the customer detail in a sequential database
- A method of output showing whether the individual seats have been booked or not
- Maintain structure of the seating plan, and resolve any potential conflicts within the customer bookings.
    - For example, two customers cannot have the same seat
- Calculations which show the total income for each performance
- Retrieve information from the customer booking database
- Be able to manipulate multiple events

Limitations

The solution should be extensive in fulfilling the aims; however every solution has its drawbacks.

- The booking database cannot be accessed or edited online, which lowers the security requirements.
- The software is limited to Windows OS
- Data structure will remain consistent across all databases

Assumptions

In order to elaborate the specification, assumptions need to be made to progress:

- The theatre seating plan does not change, because it is integral to the structure of the database

- Calculation of income which are calculated from the ticket prices and booking numbers
- The booking database should be able to be printed as another method of output.
- L15 is a disabled seat because it is at the end of a row.

Objectives

Data Storage and Manipulation

- Creation of a database for each event which will store customer booking information, which will be blank
- Records can be:
    a. added in according places because the database will sequential and input data while in the main application
    b. selected then deleted to remove records
    c. edited to make amendments and refunds
    d. retrieved to view information by either viewing the seating plan or table view
- Sorting dynamically on each heading such as names, seats in different methods such as ascending order or descending order.
- Find and replace all records
- Searching and filtering each record in the database by conditions, for example to display all available seats
- The data should store some basic information about the performance itself, such as the showing date, and the genres.
- Program will be able to cope with different formats of data so that it can widely use with other applications
- Opening and saving the database using file structures on the hard drive disk

Interfaces

- A easy to use system that are user friendly
    - Organized controls on a form
    - Easily accessible controls, using tooltips, labels and tab order
- An graphical representation on seating plan that is dynamic

  - Ability to click the box representing the seat, which will refer to the record on the seating plan
  - The each seat in the seating plan will change colour according to whether the seat has been booked or not
- Data can be printed out for hard-copy reports or booking seating plan with highlighted seats
- A login form which will used for the user to enter in the username and passwords
- About form to show program information such as version number and author.

Verification and Validation

- Data entered will have masks which will skip invalid key-presses
- Database will check whether a seat is not double booked
- Verification and validation which assures the customer booking the correct details on the correct date
- Some entries must be filled in

Calculation

- Income calculated by the sum of booked seats in the performance
- Expenditures calculated by overheads
- Profit by expenditures subtracted by the income

Security

- Password protected program by an account using hash coding to prevent program being compromised
- Data is compressed to discourage editing raw database booking data

Justification of the Proposed Solution

I have chosen Visual C++ because it offers:

- An extensive library which makes C++ suitable for a wide range of solutions. This language can also use libraries from complied from other languages such as Visual Basic. For instance, a rich GUI can be made using a wide range of controls such as buttons, textboxes and labels.
- There is a large community for C++, meaning there is vast amount of help and support.
- The language is professional and widely used, so high standards of software can be produced.

Visual Studio for Visual C++ is the most suitable SADS booking system, because it allows all of the objectives to be completed to a high standard. For example, the dynamic graphical representation of the seating plan can be constructed from the .NET Framework's graphics classes from Visual C++. Visual Studio allows the form, like the login form interfaces, to be created quickly, especially from the intuitive drag and drop feature.

External data management libraries are not required because of the .NET Framework's huge range of features. Facilities such as saving the database and retrieving the information can be achieved using advanced techniques such as saving individual fields in binary for compression. Moreover, printing can be done by using the Printing category of the .NET F**ra**mework.

Data Structure and Method of Access

Data Structure

I will have two sets of bookings table for each performance or day. The data will contain all the information for that day, including the customer information. A variable length will contain the string's length stored in one byte then the actual contents of the string. Note there is no field determining the presence of the record, since if the record does not exist, therefore the seat is not booked. In additional, all fields must be entered, as a presence check.

Field Name	Data Type	Length	Purpose	Example Data	Sorting	Searching	Validation
Seat Code	String	Variable	Primary Key, to uniquely identify records from each other	L14	Yes	Yes	Format Check that both the Seat Row and Seat Number are within range
Date Booked	Date	8 Bytes	The date the booking has been made	14/11/12	Yes	Yes	Range Check between SADS's creation and current date.
Customer Forename	String	Variable	The customer's first name	Bob	No	Yes	Type Check to prevent any numbers
Customer Surname	String	Variable	The customer's last name	Joe	Yes	No	Type Check to prevent any numbers
Customer House Number	String	Variable	The customer's house number in the address	23A	No	No	Presence Check that it exists
Customer Post Code	String	6 Bytes	The customer's post code for his/her house	B32 64P	No	No	Format Check in L00 0LL, where '0' is a number, and 'L' is a letter (Verification through masked textbox)
Customer Telephone	String	11 Bytes	The customer's contact information using telephone	0121 746 6524	No	No	Format Check in 00000000000 mask (Verification through masked textbox)

The searching will be attempted to done as efficiently as possible. Binary search will be used when the column of fields are sorted, otherwise linear search will be used. Note that searching will only select the first or next item (if the first item has already been selected), and will not filter the other records. The searching can be done in ascending order, or descending, depending on the user's commands.

The validation check will be performed when the user submits the data on the data entry form, which is typically the 'OK' button.

Note that record and booking are used interchangeably within this project.

Method of Access

The data will be accessed directly from a file on the client's computer. The program will retrieve the data when the program is started (or create a new file when the file does not exist). So when the user opens the application, all the data will be there available to the user. When the program is promoted to save, or when the program closes, the file will be updated with any changes. The data will be stored alongside the application's executable, typically on a hard drive. This means the data will be able to be retrieved and stored without loss of data of the booking system.

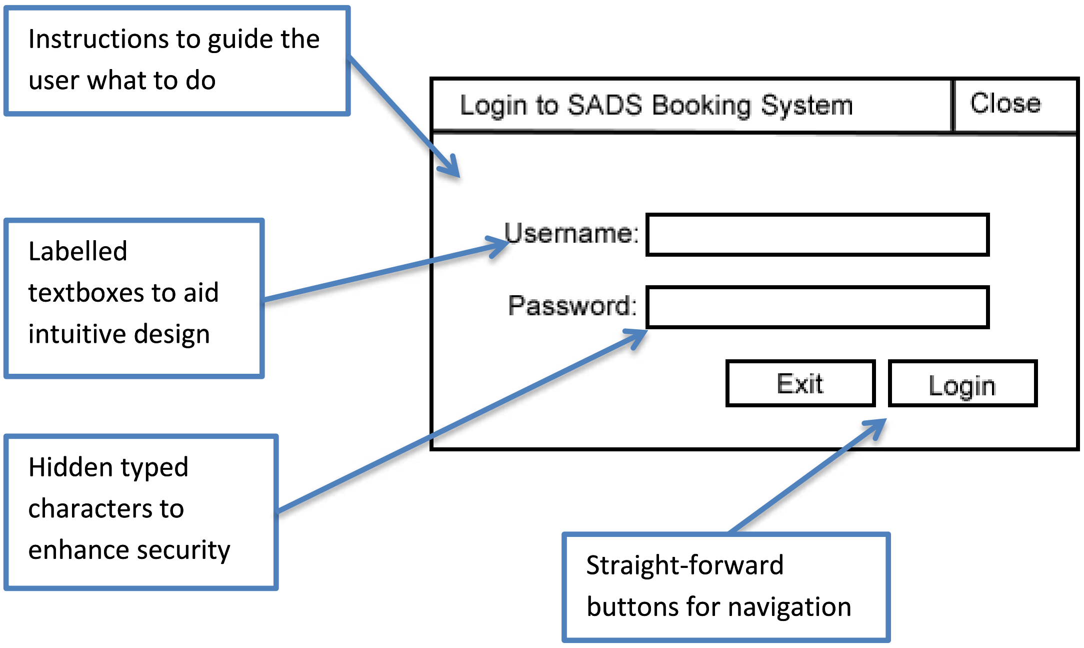
The file will be accessed using the .NET Framework's IO classes. These classes allow precision of every bit in a file, allowing a compact database to be created and modified. I will manually code how the program will read and write the file. At the start of the file, general theater information will be saved, such as performance dates. Each record will be read in a loop, going through all the possible seats. Not only this method is easy to understand, but allows the data to be highly compressed. This is an advantage because this also adds to an element of security, because the third party will have a difficult time directly modifying the file, since they do not know how it is structured.

Technical details on how the file will be read and saved, will be featured in the pseudo-code section later.

User Interface

Note that the default icon will be replaced when the application will be developed, and so these may change in development. Throughout the user interface designs, all forms have been considered for simplicity and ease of access.

Login Form



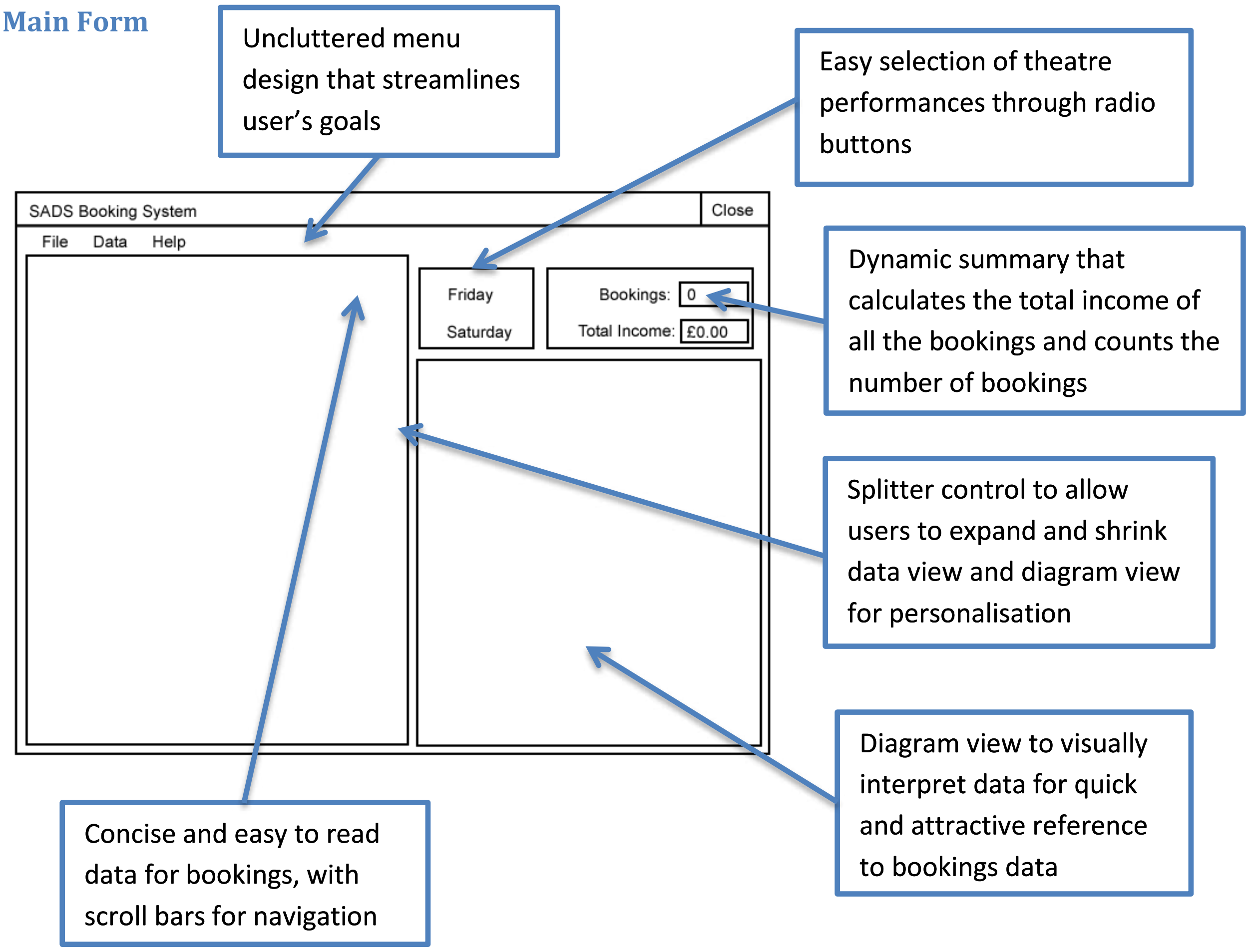
Instructions to guide the user what to do

Labelled textboxes to aid intuitive design

Hidden typed characters to enhance security

Straight-forward buttons for navigation

Main Form



Uncluttered menu design that streamlines user’s goals

Easy selection of theatre performances through radio buttons

Dynamic summary that calculates the total income of all the bookings and counts the number of bookings

Splitter control to allow users to expand and shrink data view and diagram view for personalisation

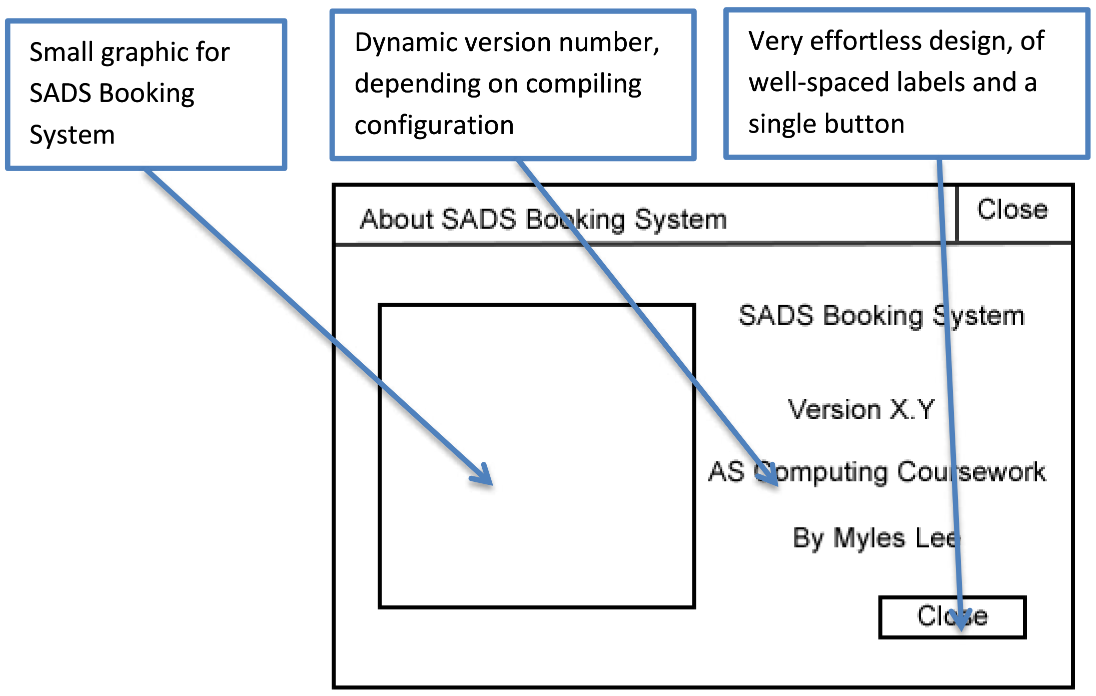
Diagram view to visually interpret data for quick and attractive reference to bookings data

Concise and easy to read data for bookings, with scroll bars for navigation

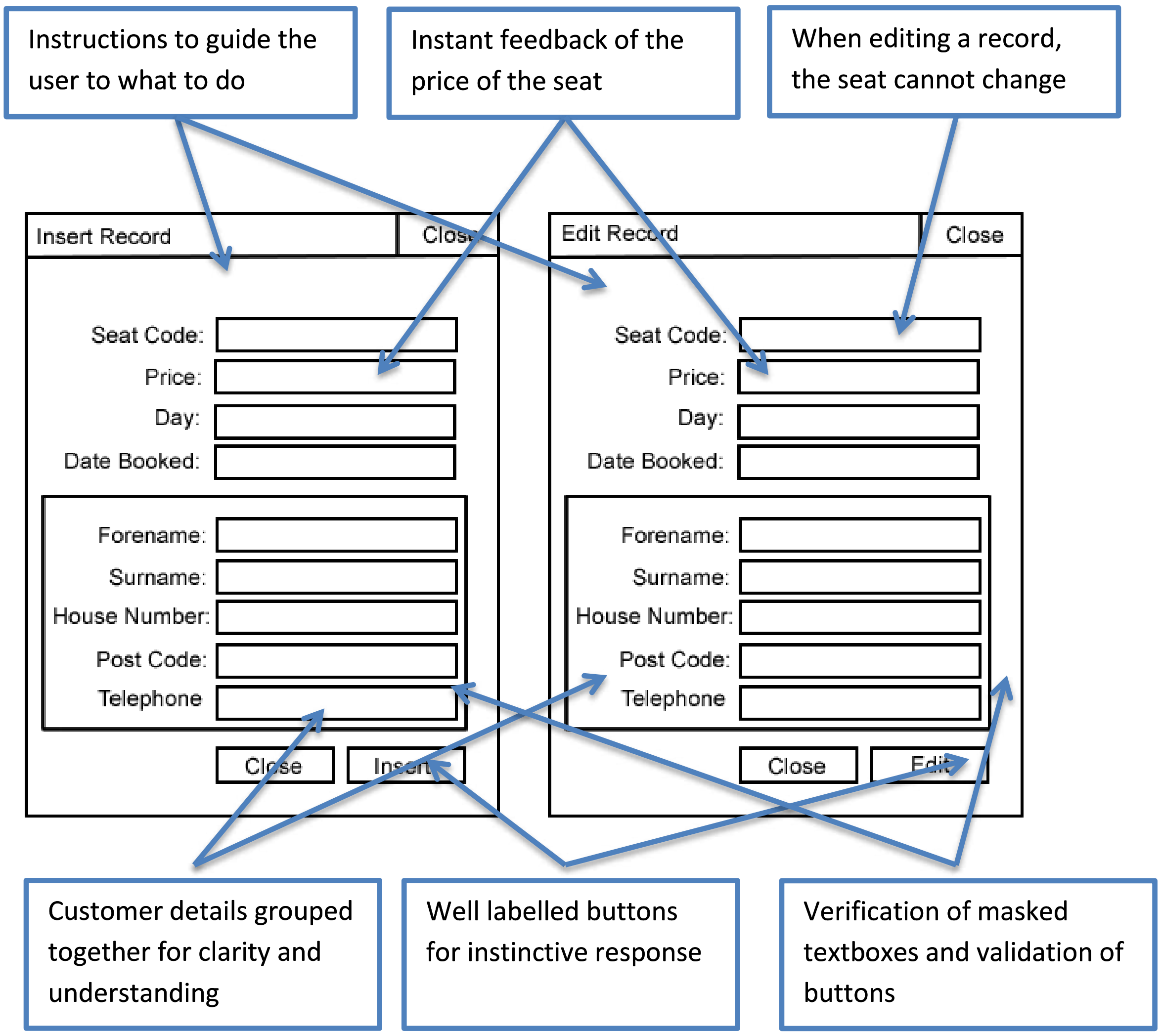
Menu Structure

File	Print Report	Prints a tabular format of the current bookings
	Print Diagram	Prints the diagram of the booking data
	Exit	Closes and exits the applicaition
Data	Add...	Creates a new bookin
	Edit...	Ammends a booking that exists
	Delete	Removes a booking from the data
	Find...	Finds a booking that matches the user's crtieras
	Sort By Seat Code	
	Sort by Date Booked	
	Sort by Forename	
Help	About...	Shows the about dialogue that holds information of SADS Booking System, such as version number

About Form


Small graphic for SADS Booking System
Dynamic version number, depending on compiling configuration
Very effortless design, of well-spaced labels and a single button
About SADS Booking System
Close
SADS Booking System
Version X.Y
AS Computing Coursework
By Myles Lee
Close

Data Entry Form


Instructions to guide the user to what to do

Instant feedback of the price of the seat

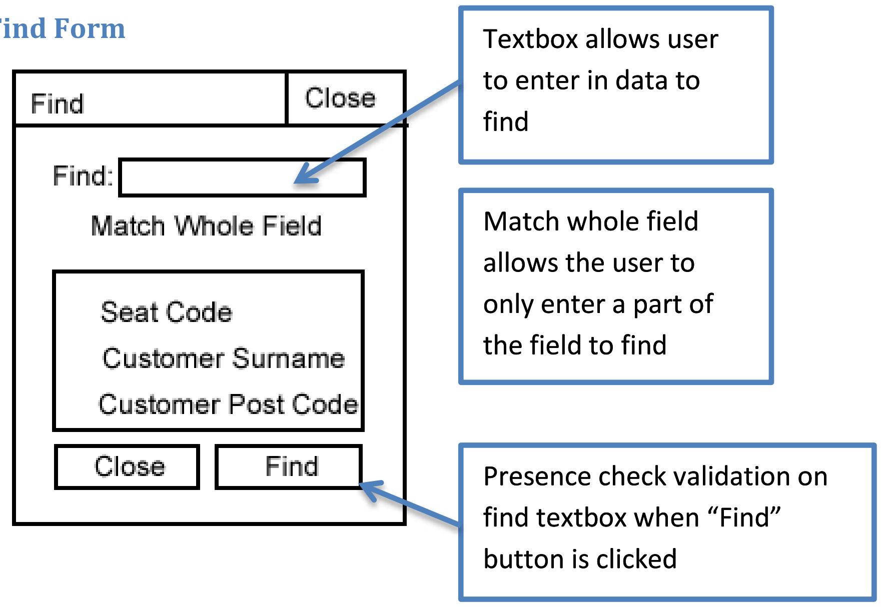
When editing a record, the seat cannot change

Customer details grouped together for clarity and understanding

Well labelled buttons for instinctive response

Verification of masked textboxes and validation of buttons

Find Form


Textbox allows user to enter in data to find

Match whole field allows the user to only enter a part of the field to find

Presence check validation on find textbox when “Find” button is clicked

Software and Hardware Requirements

Minimum Hardware Requirements

- A standard computer system
    - 256 MB of RAM to run the program
    - Basic CPU processor
- Monitor to display the GUI of the program
- Keyboard to enter text fields and optional navigation
- Mouse to select items of the GUI
- A gray scale (black and white) printer to output reports

Recommended Hardware Requirements

- A high quality printer to produce more attractive and professional reports
- A monitor size of 1024 by 768 to display the seating plan diagram and data together

Minimum Software Requirements

- Minimum Windows XP or later operating system to:
    - Run executable files
    - Provide security for the database
    - Allows easy backup of the database
- .NET Framework 3 Service Pack 1
    - Allows efficient development of the program
    - Should be pre-installed on the OS

Evaluation Criteria

Many different methods of testing will be applied on the application to produce an accurate evaluation, involving both: black box and white box testing.

Functionality and Suitability

The functionality of the program will be assessed against the objectives, which is mention earlier on in this document. Each objective would be given a suitable task to test whether it performs as expected. For example, the retrieval of the database can be checked by entering the data and restarting the application.

Alternatively, every feature can be executed to test the program's functionality. This can be done by accessing every item on the menu. Moreover, the questionnaire can also give an insight to what the user requires, which is a part of functionality. The success criteria would be that 95% or more features work perfectly.

Usability and Accessibility

Every feature would be checked that it could be accessed in more than one way. For example, both the mouse and keyboard can access the exit button. Each item should be able to be accessed through the menu, since alternative keys can be used to expand sub-menus and select the menu items. The program will be designed with programming and graphical conventions to maintain and enhance professionalism.

The accessibility can be accessed by the time it takes for potential clients to use the application. For instance, if a computer novice user spends minutes navigating through the menus, and cannot find the feature he/her requires, then the program is poorly developed.

Furthermore, the questionnaire's results will also indicate the usability of the application, by the feedback scores. The success criteria are that all features are easy to access, by all computer novice users.

Performance

In conjunction with the functionality, each feature would be tested for the time taken. If the process, such as saving the file takes too long, then the program is inefficient. The actual time taken is irrelevant, since the perceived performance is what users judge the program against. Therefore, the questionnaire would provide good feedback.

More technically, strict memory tests can performed on the application. For example, the memory allocation of the application can be monitored, to check whether the application has a memory leak. The performance needs to be measured so that the minimum requirements of the software can be managed by the user's computer.

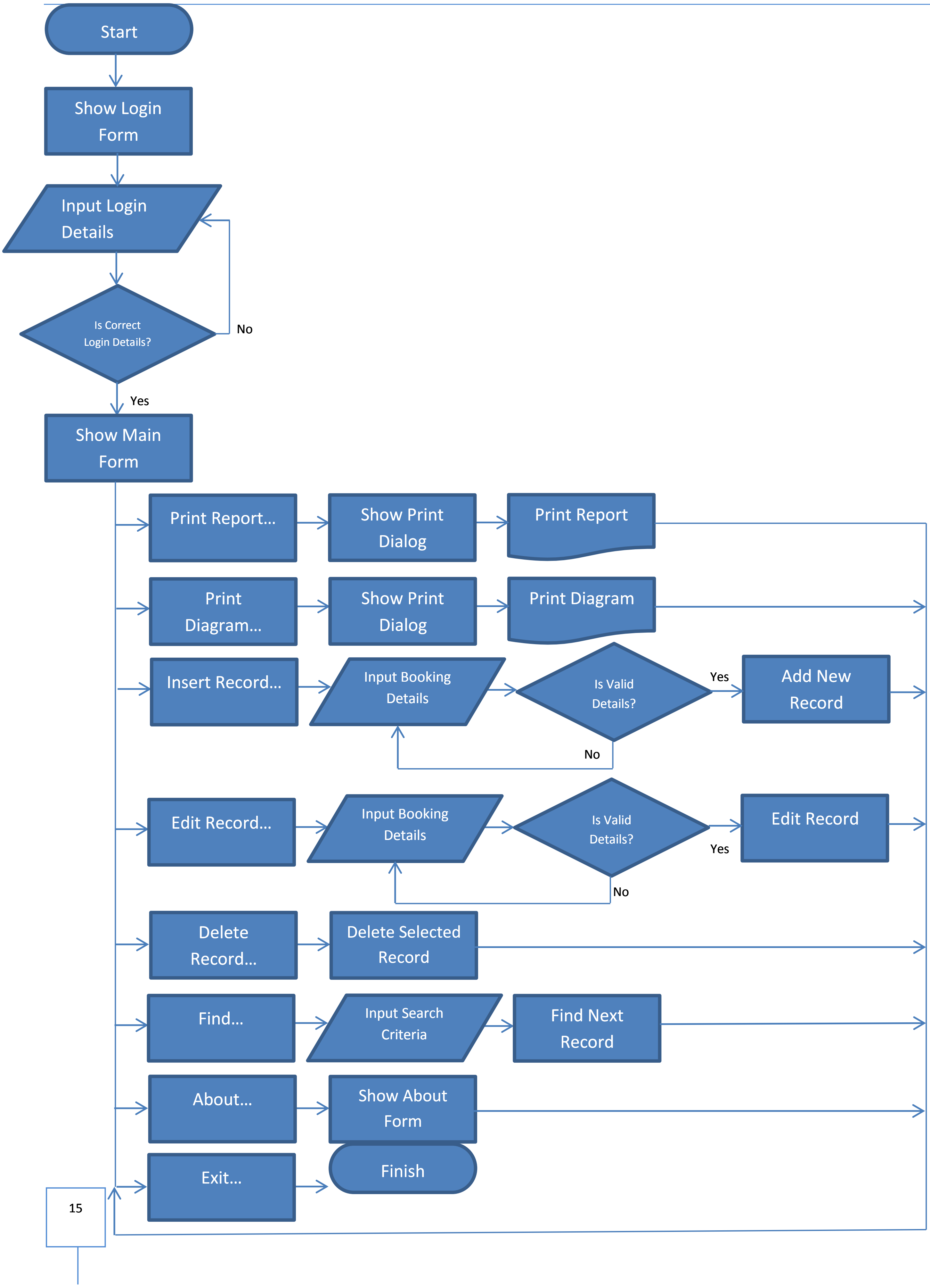
The success criteria are that all processes take less than one second.

Questionnaire

Please tick the appropriate box, depending on the views of the program.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The graphical interface is appealing.					
The program is easy to navigate.					
The program effectively manages data.					
The reports produced are clear.					
The program is robust when dealing with unexpected data.					

Please write any further comments to improve your response. Comments would be appreciated.

System Design Flow Chart



Pseudo-code

Login Form

```
PROCEDURE LoginClick
    IF Username = "admin" AND HASH Password = "JDk6QFR21ZTdE4..."
        HIDE LoginForm
        SHOW MainForm
    ELSE
        SHOW MessageBox "The username or password is incorrect"
    ENDIF
END PROCEDURE

PROCEDURE ExitClick
    EXIT Application
END PROCEDURE
```

Main Form

```
PROCEDURE FormLoad
    CALL PROCEDURE LoadData "...\Friday.sadsdb"
    CALL PROCEDURE UpdateSummary
    CALL PROCEDURE DrawDiagram
END PROCEDURE

PROCEDURE FormClose
    IF RadioButtonFriday IS CHECKED
        CALL PROCEDURE SaveData "...\Friday.sadsdb"
    ELSE
        CALL PROCEDURE SaveData "...\Saturday.sadsdb"
```

```
     ENDIF

     EXIT Application
END PROCEDURE

PROCEDURE DayChanged
     IF RadioButtonFriday IS CHECKED
          CALL PROCEDURE SaveData "...\Saturday.sadsdb"
          CALL PROCEDURE LoadData "...\Friday.sadsdb"
     ELSE
          CALL PROCEDURE SaveData "...\Friday.sadsdb"
          CALL PROCEDURE LoadData "...\Saturday.sadsdb"
     ENDIF

     CALL PROCEDURE UpdateSummary
     CALL PROCEDURE DrawDiagram
END PROCEDURE

PROCEDURE DiagramSizeChanged
     CALL PROCEDURE DrawDiagram
END PROCEDURE

PROCEDURE Exit
     CALL PROCEDURE FormClose
END PROCEDURE

PROCEDURE Insert
     Repeat AS Boolean
     DataEntryForm.InsertForm = TRUE
```

```
    IF RadioButtonFriday IS CHECKED
        DataEntryForm.Day = "Friday"
    ELSE
        DataEntryForm.Day = "Saturday"
    ENDIF

    REPEAT
        DataEntryForm.SeatCode = Empty
        SHOW DataEntryForm
        Repeat = CALL PROCEDURE CheckSeat DataEntryForm.SeatCode AND DataEntryForm.SeatCode NOT Empty

        IF Repeat
            SHOW MessageBox "The seat has already been booked."
        ENDIF
    WHILE Repeat

    IF DataEntryForm.SeatCode NOT Empty
        NEW Record
        CALL PROCEDURE UpdateSummary
        CALL PROCEDURE DrawDiagram
    ENDIF
END PROCEDURE
```

PROCEDURE Edit

```
    IF RecordCount = 0
        SHOW MessageBox "Please select a record with data."
    ELSE
        Repeat AS Boolean
        DataEntryForm.InsertForm = FALSE
```

```
          IF RadioButtonFriday IS CHECKED
               DataEntryForm.Day = “Friday”
          ELSE
               DataEntryForm.Day = “Saturday”
          ENDIF

          DataEntryForm.SeatCode = CurrentRow.SeatCode
          DataEntryForm.DateBooked = CurrentRow.DateBooked
          DataEntryForm.Forename = CurrentRow.Forename
          DataEntryForm.Surname = CurrentRow.Surname
          DataEntryForm.HouseNumber = CurrentRow.HouseNumber
          DataEntryForm.PostCode = CurrentRow.PostCode
          DataEntryTelephone = CurrentRow.Telephone

          REPEAT
               SHOW DataEntryForm
          Repeat = CALL PROCEDURE CheckSeat DataEntryForm.SeatCode AND NOT
                    DataEntryForm.SeatCode = Empty AND
                    DataEntryForm.SeatCode = CurrentRow.SeatCode

          IF Repeat
               SHOW MessageBox “The seat has already been booked.”
          ENDIF
     WHILE Repeat

     IF NOT DataFormEntry.SeatCode = Empty
          EDIT Record
          CALL PROCEDURE UpdateSummary
```

```
        CALL PROCEDURE DrawDiagram
    END IF
END PROCEDURE
```

PROCEDURE Delete

```
    IF RowCount = 0
        SHOW MessageBox "Please select a record with data."
    ELSE
        DELETE Record
        CALL PROCEDURE UpdateSummary
        CALL PROCEDURE DrawDiagram
    ENDIF
END PROCEDURE
```

PROCEDURE Find

```
    SHOW FindForm

    IF NOT FindForm.SearchValue = Empty

        i AS Integer

        IF RowCount = 0
            SHOW MessageBox "There are no records to search."
        ELSE
            StartRow AS Integer

            IF CurrentRow IS Valid Find
                StartRow = CurrentRow.Index + 1
            ENDIF
        FOR i = StartRow TO RowCount STEP 1
            IF Row[i] IS Valid Find
                SELECT Row[i]
                EXIT FUNCTION
            END IF
        ENDFOR

        SHOW MessageBox “No more records are found.”
    ENDIF
END PROCEDURE
```

PROCEDURE About

```
    SHOW AboutForm
END PROCEDURE
```

PROCEDURE PrintReportDocument

```
    Page AS Integer = 0
    Row AS Integer = 0
    MaxRows AS Integer = 53
    i AS Integer

    IF Page = 0
        MaxRows = 47

        DRAW Text Title
        DRAW Text Day
        DRAW Text NumberofBookings
        DRAW Text TotalIncome
```

```
     ENDIF

     DRAW Text RowHeaders
     DRAW Line Below RowHeaders

     FOR i = 0 TO MaxRows STEP 1
          IF Row < RowCount
               DRAW Text Record
               Row = Row + 1
               DRAW Line Below Record
          ELSE
               Page = 0
               Row = 0
               EXIT FUNCTION
          ENDIF
     ENDFOR

     IF Row < RowCount
          MorePages = TRUE
          Page = Page + 1
     ELSE
          Page = 0
          Row = 0
     ENDIF
END PROCEDURE
```

PROCEDURE PrintDiagramDocument

```
     DRAW CALL PROCEDURE DrawDiagram
END PROCEDURE
```

```
PROCEDURE PrintReportClick
      SHOW PrintDialog
END PROCEDURE

PROCEDURE PrintDiagramClick
      SHOW PrintDialog
END PROCEDURE

PROCEDURE SaveData
      INPUT FilePath
      OPEN File AT FilePath

      i AS Integer

      FOR i = 0 TO RowCount STEP 1
            WRITE SeatCode
            WRITE BookingDate
            WRITE Forename
            WRITE Surname
            WRITE HouseNumber
            WRITE PostCode
            WRITE Telephone
      ENDFOR

      WRITE "END"

      CLOSE File
END PROCEDURE
```

PROCEDURE LoadData

```
    INPUT FilePath

    IF NOT File Exist At FilePath
        EXIT FUNCTION
    ENDIF

    OPEN File
    CLEAR Records
    i AS Integer

    FOR i = 0 TO 196 STEP 1
        READ SeatCode

        IF SeatCode = "END"
            EXIT FOR
        ELSE
            NEW Record
            READ SeatCode
            READ DateBooked
            READ Forename
            READ Surname
            READ HouseNumber
            READ PostCode
            READ Telephone
        ENDIF
    ENDFOR

    CLOSE File
```

```
END PROCEDURE
```

PROCEDURE UpdateSummary

```
      NumberofBooks = RowCount

      SeatRow AS Character
      TotalIncome AS Float = 0
      i AS Integer

      FOR i = 0 TO RowCount STEP 1
            SeatRow = SeatCode[0]

            IF SeatRow = "C"
                  TotalIncome = TotalIncome + 10.0
            ELSE IF SeatRow = "F"
                  TotalIncome = TotalIncome + 12.5
            ELSE
                  TotalIncome = TotalIncome + 7.25
            ENDIF
      ENDFOR
END PROCEDURE
```

PROCEDURE DrawDiagram

```
      SeatRowCount AS Integer = 11
      GridWidth AS Float = Width / 24.0
      SeatRow AS Character
      SeatNumber AS Integer
      CurrentGrid AS Point = (0, 0)
      i AS Integer
```

```
NEW Bitmap

DRAW Text Title

FOR i = 0 TO RowCount STEP 1
     IF i = 0
          SeatRow = L
          SeatNumber = 15
          CurrentGrid.X = GridWidth * 6.0
     ELSE IF i = 1
          SeatRow = K
          SeatNumber = 19
          CurrentGrid.X = GridWidth * 2.0
     ELSE IF i = 2
          SeatRow = J
          SeatNumber = 19
          CurrentGrid.X = GridWidth
     ELSE IF i = 3
          SeatRow = H
          SeatNumber = 19
          CurrentGrid.X = GridWidth
     ELSE IF i = 4
          SeatRow = G
          SeatNumber = 19
          CurrentGrid.X = GridWidth
     ELSE IF i = 5
          SeatRow = F
          SeatNumber = 20
```

```
    CurrentGrid.X = GridWidth
ELSE IF i = 6
    SeatRow = E
    SeatNumber = 20
    CurrentGrid.X = GridWidth
ELSE IF i = 7
    SeatRow = D
    SeatNumber = 19
    CurrentGrid.X = GridWidth
ELSE IF i = 8
    SeatRow = C
    SeatNumber = 17
    CurrentGrid.X = GridWidth * 2.0
ELSE IF i = 9
    SeatRow = B
    SeatNumber = 16
    CurrentGrid.X = GridWidth * 2.0
ELSE IF i = 10
    SeatRow = A
    SeatNumber = 14
    CurrentGrid.X = GridWidth * 2.0
ENDIF

CurrrentGrid.Y = GridWidth * (i + 3)

FOR SeatNumber TO 0 STEP -1
    IF CurrentGrid.X = GridWidth * 6.0
        DRAW Text RowChar
        CurrentGrid.X = CurrentGrid.X + GridWidth * 2
```

```
                ENDIF

                IF CheckSeat SeatCode
                     DRAW Rectangle Fill LightGray
                ENDIF

                DRAW Rectangle Border
                DRAW Text SeatNumber

                CurrentGrid.X = CurrentGrid.X + GridWidth
           ENDFOR
      ENDFOR

      DRAW Lines Stage
      DRAW Text Stage

      DRAW Lines Key
      DRAW Text Key

      DRAW Table TicketPrices
      DRAW Text TicketPrices
END PROCEDURE
```

PROCEDURE CheckSeat

```
      INPUT SeatCode
      i AS Integer
      FOR i = 0 TO RowCount STEP 1
           IF Rows[i].SeatCode = SeatCode
                RETURN TRUE
```

```
            ENDIF
      ENDIF
      RETURN FALSE
END PROCEDURE
```

Data Entry Form

PROCEDURE IsText

```
      INPUT Value
      i AS Integer
      FOR i = 0 TO Value.Length STEP 1
            IF Value[i] Is Digit
                  RETURN FALSE
            ENDIF
      ENDFOR
      RETURN TRUE
END PROCEDURE
```

PROCEDURE FormLoad

```
      DayLabel = Day

      IF NOT InsertForm
            Title = "Edit Record"
            Caption = "Please ensure changes are accurate."
            Button = "Edit"


            txtSeatCode = SeatCode
            dtpDate = DateBooked
            txtForename = Forename
            txtSurname = Surname
            txtHouseNumber = HouseNumber
            txtPostCode = PostCode
            txtTelephone = Telephone
      ENDIF
END PROCEDURE
```

PROCEDURE AcceptClick

```
      SeatRow AS Character = SeatCode[0]
      SeatNumber AS Integer = SeatCode[1 to 2]

      IF SeatNumber = 0
            SHOW MessageBox "Please enter a valid seat number."
      ELSE IF SeatRow IS Out of Range
            SHOW MessageBox "Please enter a valid seat row."
      ELSE IF SeatNumber IS Out of Range
            SHOW MessageBox "Please enter a valid seat number."
      ELSE IF Date < 1/1/1996 OR Date > Now
            SHOW MessageBox "Please enter a valid booking date."
      ELSE IF Forename = Empty OR NOT IsText Forename
            SHOW MessageBox "Please enter a valid forename."
      ELSE IF Surname = Empty OR NOT IsText Surname
            SHOW MessageBox "Please enter a valid surname."
      ELSE IF HouseNumber = 0
            SHOW MessageBox "Please enter a valid house number."
      ELSE IF PostCode.Length < 6
            SHOW MessageBox "Post code is too short."
      ELSE IF Telephone.Length < 7
```

```
            SHOW MessageBox "Telephone number is too short."
      ELSE
            SeatCode = txtSeatCode
            DateBooked = dtpDate
            Forename = txtForename
            Surname = txtSurname
            HouseNumber = txtHouseNumber
            PostCode = txtPostCode
            Telephone = txtTelephone
            CLOSE AboutForm
      ENDIF
END PROCEDURE
```

PROCEDURE SeatCodeChanged

```
      SeatRow AS Character = SeatCode[0]

      IF SeatRow <= "C"
            SeatPrice = "£10.00"
      ELSE IF SeatRow <= "F"
            SeatPrice = "£12.50"
      ELSE IF SeatRow <= "L" AND NOT SeatRow = "I"
            SeatPrice = "£7.25"
      ELSE
            SeatPrice = "Unknown"
      ENDIF
END PROCEDURE
```

Find Form

```
PROCEDURE FormLoad
    RESTORE Control Properties
END PROCEDURE

PROCEDURE FindClick
    IF SearchValue = Empty
        SHOW MessageBox "Please enter in what to find"
    ELSE
        STORE Control Properties
        CLOSE FindForm
    ENDIF
END PROCEDURE

PROCEDURE CloseClick
    SearchValue = Empty
END PROCEDURE
PROCEDURE FormClose
    STORE Control Properties
END PROCEDURE
```

Program Documentation

Data Structure and Variables

The data structure has not changed, and is identical to the table in the Data Structure section above. Note that only variables within functions will be mentioned, since controls and form variables will make the table lengthy. Their code can be found in a later stage in this document.

Main Form

Variable	Data Type	Scope	Procedure	Usage
Diagram	Bitmap	Global	N/A	Stores the bitmap that is used to generate and display the seating plan
Repeat	Boolean	Local	Insert	Stores whether to show the data entry form again
DataForm	Data Entry Form	Local	Insert	Stores the form that is used to collect the booking data from the user
Cells	Array of Objects	Local	Insert	Stores the data collected from the data entry form and encapsulates it into a single variable
Repeat	Boolean	Local	Edit	Stores whether to show the data entry form again
DataForm	Data Entry Form	Local	Edit	Stores the form that is used to collect the booking data from the user
FindForm	Find Form	Local	Find	Stores the form that is used collect the search criteria data
AboutForm	About Form	Local	About	Stores the form that displays the information about the software.
e	PrintPageEventArgs	Local	PrintReportDocument	Stores information about the current printing progress.
Page	Integer	Local	PrintReportDocument	Stores the current printing page index

Row	Integer	Local	PrintReportDocument	Stores the index of the selected booking
MaxRows	Integer	Local	PrintReportDocument	Stores the maximum number of bookings that can be printed on one page.
HeadingY	Float	Local	PrintReportDocument	Stores the Y location of the table headers.
Heading	Font	Local	PrintReportDocument	Stores the font that is used for the table headers.
Body	Font	Local	PrintReportDocument	Stores the font that is used for the table main contents.
TitleFont	Font	Local	PrintReportDoucment	Stores the font that is used for the documents title.
TitleText	String	Local	PrintReportDocument	Stores the text that is displayed as a title.
TitleSize	SizeF	Local	PrintReportDocument	Stores the pixel size of the document title
e	PrintPageEventArgs	Local	PrintDiagramDocument	Stores information about the current printing progress.
FilePath	String	Local	SaveData	Stores the location of the file to save to.
FileStream	Stream	Local	SaveData	Stores the handle to access to files.
Writer	BinaryWriter	Local	SaveData	Stores the methods that are used to writer files onto the user’s storage device.
i	Integer	Local	SaveData	Stores the loop counter for each record.
FilePath	String	Local	LoadData	Stores the location of the file to save to.
FileStream	Stream	Local	LoadData	Stores the handle to access to files.
Writer	BinaryWriter	Local	LoadData	Stores the methods that are used to writer files onto the user’s storage device.

i	Integer	Local	LoadData	Stores the loop counter for each record through the data summary.
SeatCode	String	Local	LoadData	Stores the selected record’s seat code.
SeatRow	Character	Local	UpdateSummary	Stores the selected record’s seat row
TotalIncome	Float	Local	UpdateSummary	Stores the total income for all the records on that day.
i	Integer	Local	UpdateSummary	Stores the loop counter for each record through the data summary.
Width	Integer	Local	DrawDiagram	Stores the width of the bitmap that contains the diagram.
SeatRowCount	Integer	Local	DrawDiagram	Stores the number of rows in the seating plan.
SeatRow	Character	Local	DrawDiagram	Stores the seat row for that selected seat.
SeatNumber	Integer	Local	DrawDiagram	Stores the seat number for that selected seat.
CurrentGrid	PointF	Local	DrawDiagram	Stores the current drawing coordinates of the current seat on the diagram.
Paint	Graphics	Local	DrawDiagram	Stores the graphics class that is used to draw onto the diagram bitmap using methods exposed by this class.
DisabledBrush	HatchBrush	Local	DrawDiagram	Stores the brush that is used to overlay seats to mark that is available to disabled persons.
SeatFont	Font	Local	DrawDiagram	Stores the font that is used to display the seat code.
SeatRowFont	Font	Local	DrawDiagram	Stores the font that is used to show what row each row in the seating plan is.
TicketFont	Font	Local	DrawDiagram	Stores the font that is used to display ticket prices, inside the table.
SeatFormat	StringFormat	Local	DrawDiagram	Stores the arrangement of a centralized text in a rectangle.

TitleFont	Font	Local	DrawDiagram	Stores the font that is used displaying the title of the diagram.
TitleText	String	Local	DrawDiagram	Stores the text that is used when displaying the title.
TitleSize	SizeF	Local	DrawDiagram	Stores the size that the title occupies when displaying that text.
i	Integer	Local	DrawDiagram	Stores the loop counter to go through each row in the seating plan.
ThickLine	Pen	Local	DrawDiagram	Stores the pen that is used to draw the stage, using thick lines.
StageText	String	Local	DrawDiagram	Stores the text that is used when labeling the stage.
StageFont	Font	Local	DrawDiagram	Stores the font that is used when displaying the stage label.
StageSize	Font	Local	DrawDiagram	Stores the size that the string will occupy when the text is drawn onto the bitmap.
SeatCode	String	Local	CheckSeat	Stores the seat code to check whether it exists within the table or not.
i	Integer	Local	CheckSeat	Stores the loop counter to go through every record in the current table.

Login Form

Variable	Data Type	Scope	Procedure	Usage
Encrypter	SHA512	Local	LoginClick	Stores the functions that are required to hash the password.

Data Entry Form

Variable Name	Data Type	Scope	Procedure	Usage
InsertForm	Boolean	Global	N/A	Stores whether the data entry form acts as an insert form or an edit form.

SeatCode	String	Global	N/A	Stores the seat that is currently being inserted or edited.
DateBooked	DateTime	Global	N/A	Stores the date booked of the current record
Forename	String	Global	N/A	Stores the customer’s forename of the current record.
Surname	String	Global	N/A	Stores the customer’s surname of the current record.
HouseNumber	String	Global	N/A	Stores the customer’s house number of the current record.
PostCode	String	Global	N/A	Stores the customer’s post code of the current record.
Telephone	String	Global	N/A	Stores the customer’s telephone of the current record.
Day	String	Global	N/A	Stores which day database is being modified.
i	Integer	Local	IsText	Stores the loop counter to access each character in the string.
SeatRow	Character	Local	AcceptClick	Stores the extracted seat row of the selected seat.
SeatNumber	Integer	Local	AcceptClick	Stores the extracted seat number of the selected seat.
SeatRow	Character	Local	SeatCodeChanged	Stores the extracted seat row of the selected seat.
SeatRow	Character	Local	IsSeatDisabled	Stores the extracted seat row of the selected seat.
SeatNumber	Integer	Local	IsSeatDisabled	Stores the extracted seat number of the selected seat.

Find Form

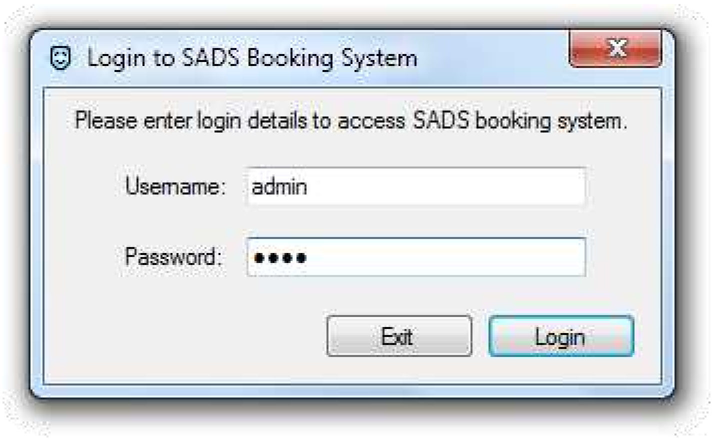
Variable	Data Type	Scope	Procedure	Usage

SearchValue	String	Global	N/A	Stores the value to search within the database.
SearchCellIndex	Integer	Global	N/A	Stores which column to search within the database.
SearchMatchWholeField	Boolean	Global	N/A	Stores whether to search whether something is exact, or contains the search field.

The final solution has the same table format. This includes a table for each day; therefore there is two Customer Bookings for each day.

User Interface

The login form, which is the first form shown to the user.



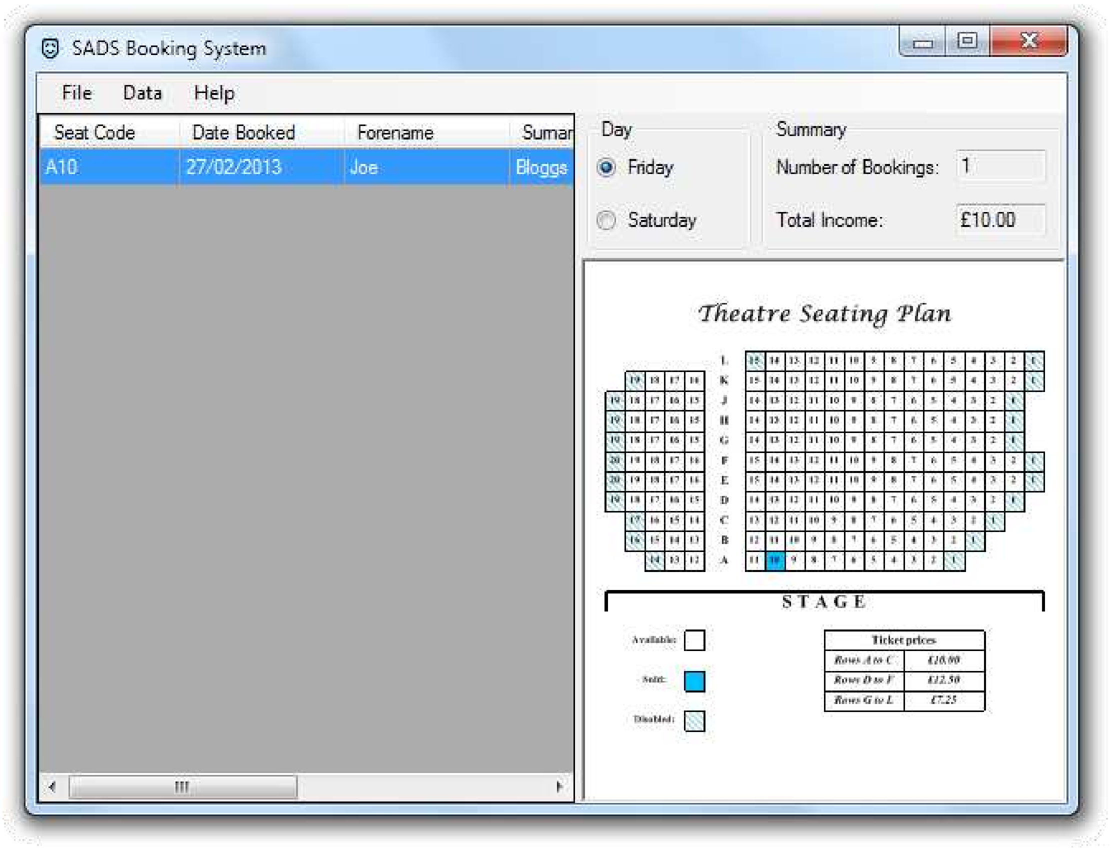
Login to SADS Booking System

Please enter login details to access SADS booking system.

Username: admin

Password: ●●●●

The main form, which shows the interface that, allows the user to interact with the bookings databases.



SADS Booking System

Seat Code	Date Booked	Forename	Sumar
A10	27/02/2013	Joe	Bloggs

Day: [✓ Friday] Saturday

Summary

Number of Bookings: 1

Total Income: £10.00

Theatre Seating Plan

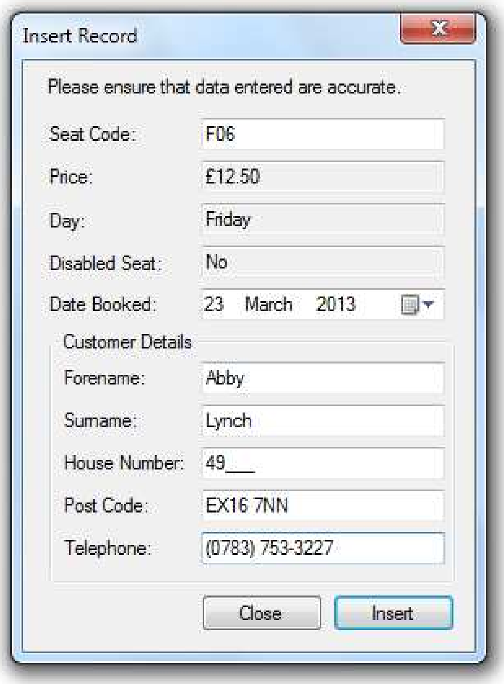
Available: ☐

Sold: 🟦

Disabled: ▨

Ticket prices	
Rows A to C	*£10.00*
Rows D to F	*£12.50*
Rows G to L	*£7.25*

The Insert Form and Edit Form are used interchangeably to create or edit new records for the selected database.

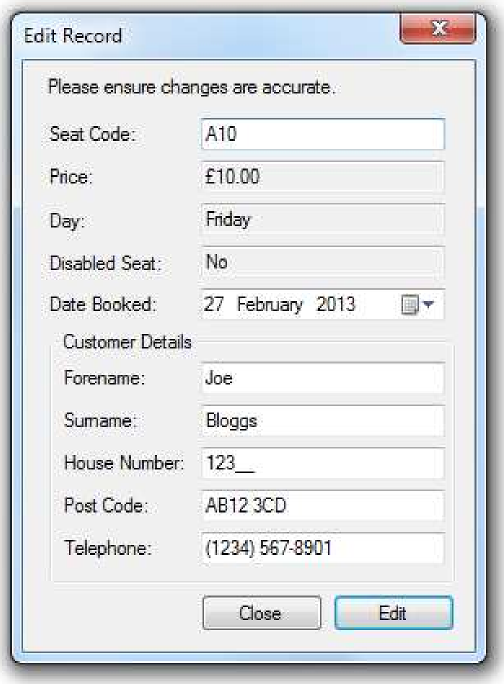

Insert Record

Please ensure that data entered are accurate.

Seat Code:	F06
Price:	£12.50
Day:	Friday
Disabled Seat:	No
Date Booked:	23 March 2013

Customer Details

Forename:	Abby
Surname:	Lynch
House Number:	49___
Post Code:	EX16 7NN
Telephone:	(0783) 753-3227


Edit Record

Please ensure changes are accurate.

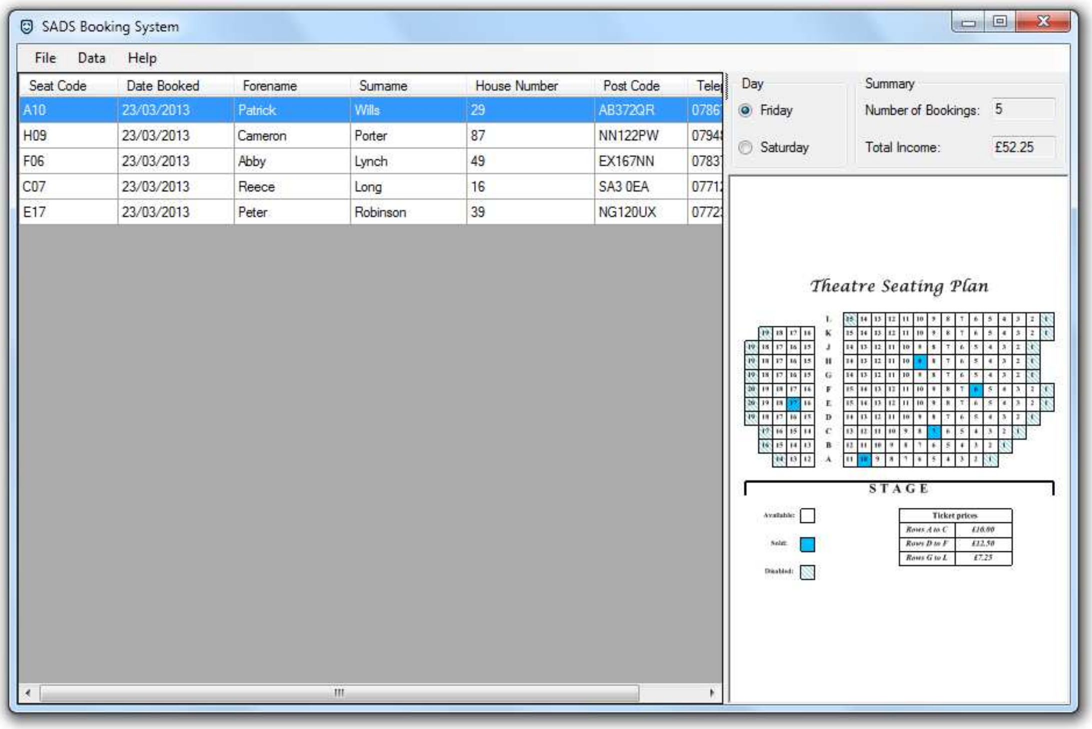
Seat Code:	A10
Price:	£10.00
Day:	Friday
Disabled Seat:	No
Date Booked:	27 February 2013

Customer Details

Forename:	Joe
Surname:	Bloggs
House Number:	123__
Post Code:	AB12 3CD
Telephone:	(1234) 567-8901

After various data has been entered, the main form updates in many ways:

- New records appear on the data grid view.
- The diagam updates to any seat changes.
- The summary box in the top right, displaying total number of records and income.


SADS Booking System

Seat Code	Date Booked	Forename	Surname	House Number	Post Code	Tele
A10	23/03/2013	Patrick	Wills	29	AB372QR	0786
H09	23/03/2013	Cameron	Porter	87	NN122PW	0794
F06	23/03/2013	Abby	Lynch	49	EX167NN	0783
C07	23/03/2013	Reece	Long	16	SA3 0EA	0771
E17	23/03/2013	Peter	Robinson	39	NG120UX	0772

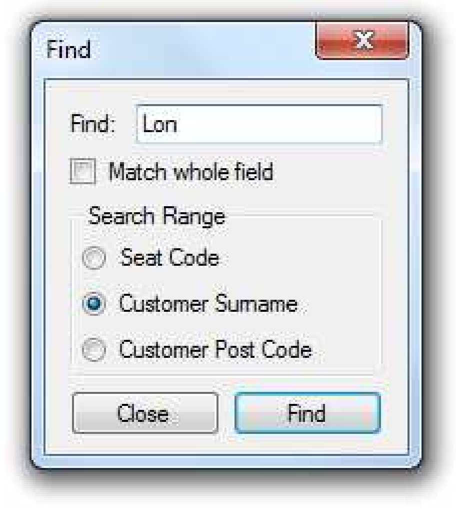
Day: [✓ Friday] Saturday

Summary

Number of Bookings:	5
Total Income:	£52.25

Ticket prices	
Rows A to C	£10.00
Rows D to F	£12.50
Rows G to L	£7.25

The Find form allows the user to locate a record within the selected database.

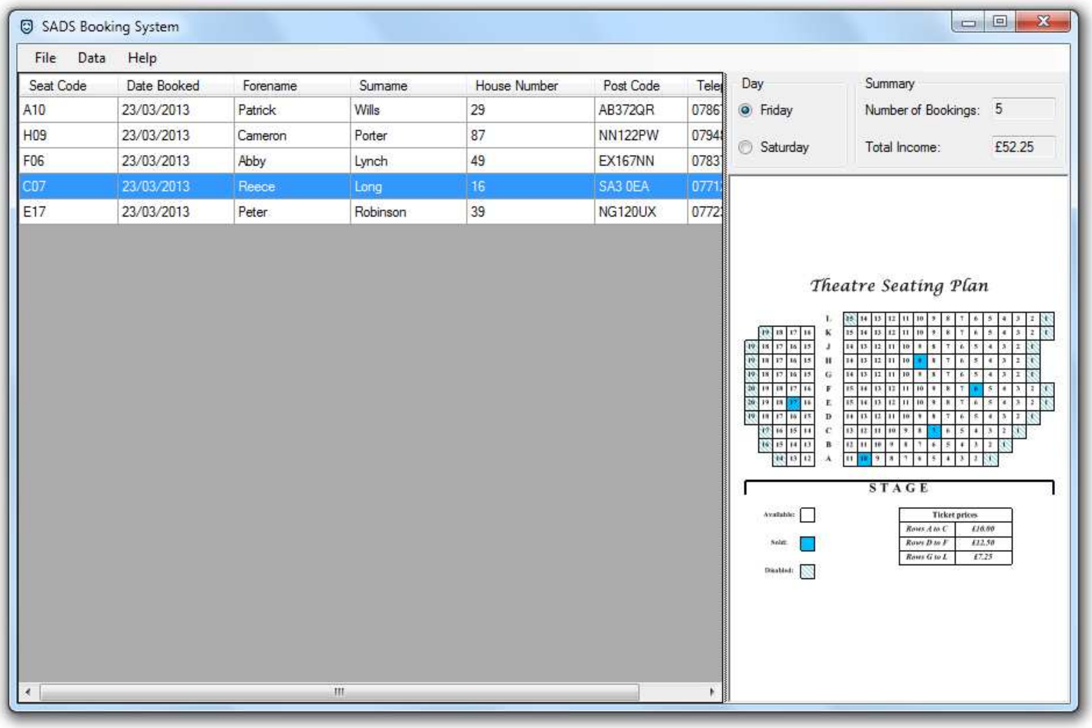


Find: Lon

[ ] Match whole field

Search Range
- Seat Code
- [✓ Customer Surname]
- Customer Post Code

The next row that matches the criteria specified by the Find form is selected. If there is no next record that matches, a message box appears, saying no record is found.



Seat Code	Date Booked	Forename	Surname	House Number	Post Code	Tele
A10	23/03/2013	Patrick	Wills	29	AB372QR	0786
H09	23/03/2013	Cameron	Porter	87	NN122PW	0794
F06	23/03/2013	Abby	Lynch	49	EX167NN	0783
C07	23/03/2013	Reece	Long	16	SA3 0EA	0771
E17	23/03/2013	Peter	Robinson	39	NG120UX	0772

Day
- [✓ Friday]
- Saturday

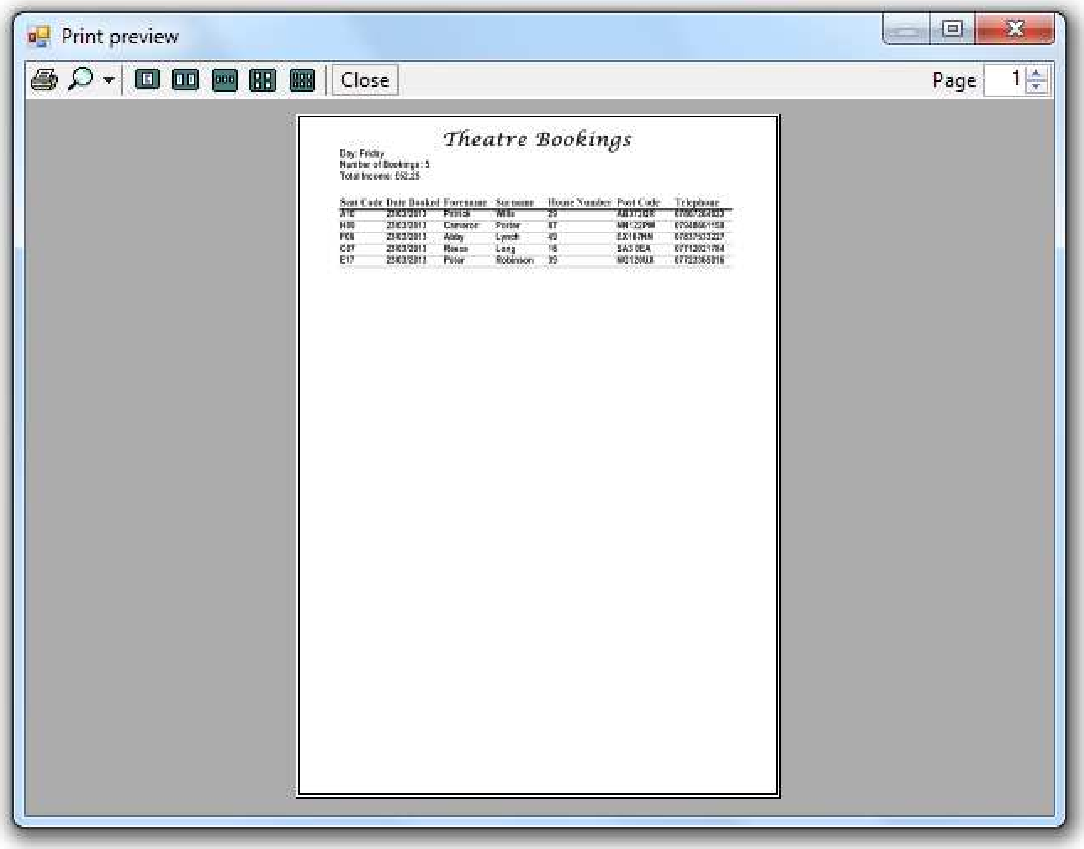
Summary

Number of Bookings: 5

Total Income: £52.25

Ticket prices	
Rows A to C	£10.00
Rows D to F	£12.50
Rows G to L	£7.25

The data from the database is reorganized to produce a report. The summary of the day is printed on the top of the page, but this only appears on the first page.

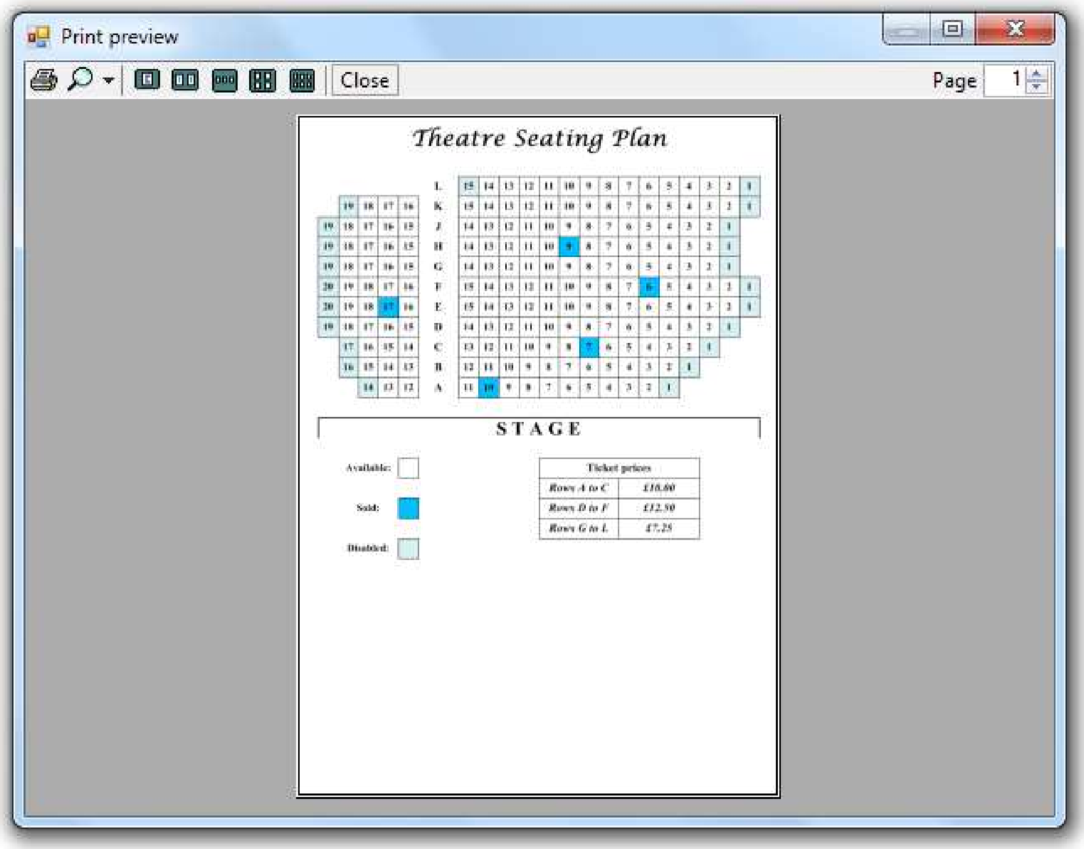


Theatre Bookings

Day: Friday
Number of Bookings: 5
Total Income: £52.25

Seat Code	Date Booked	Forename	Surname	House Number	Post Code	Telephone
A10	22/03/2013	Patrick	Willis	29	[illegible]	[illegible]
H09	22/03/2013	Cameron	Porter	87	MK122PW	07948601158
F06	22/03/2013	Abby	Lynch	40	EX167NM	07837533227
C07	22/03/2013	Reece	Long	16	SA3 0EA	07712021784
E17	22/03/2013	Peter	Robinson	39	[illegible]	07723365016

The diagram can also be printed, which is identical to the image displayed on the main form.



Theatre Seating Plan

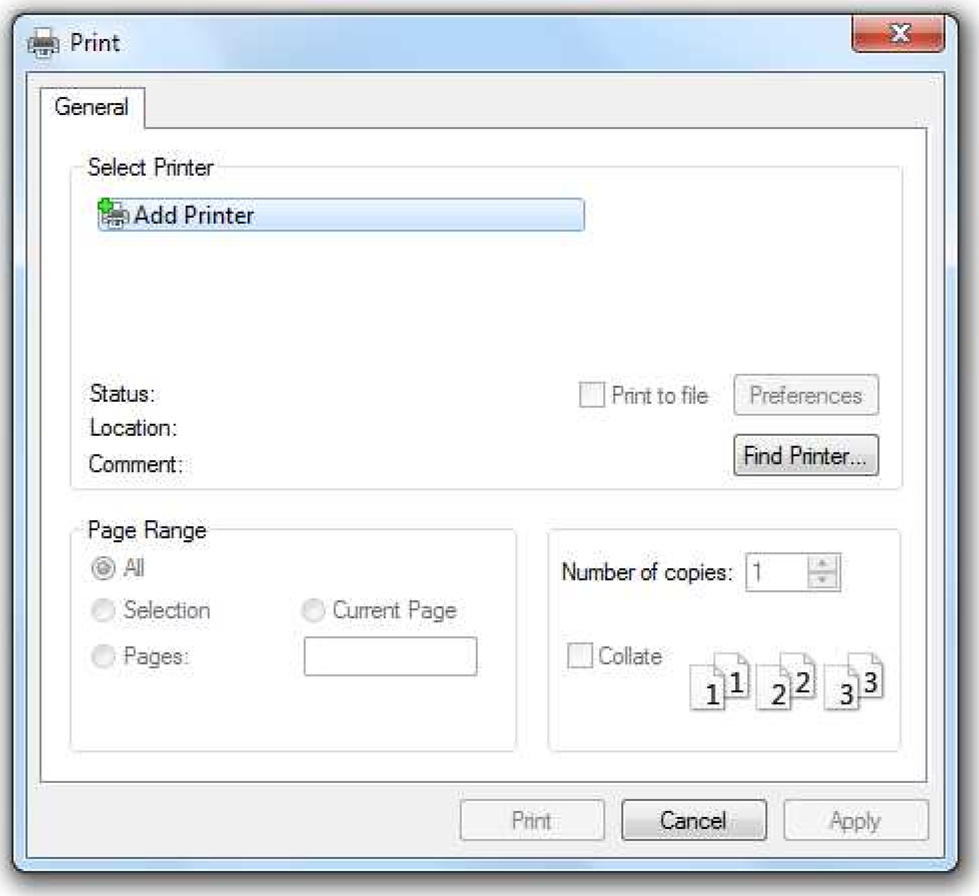
Available: ⬜

Sold: 🟦

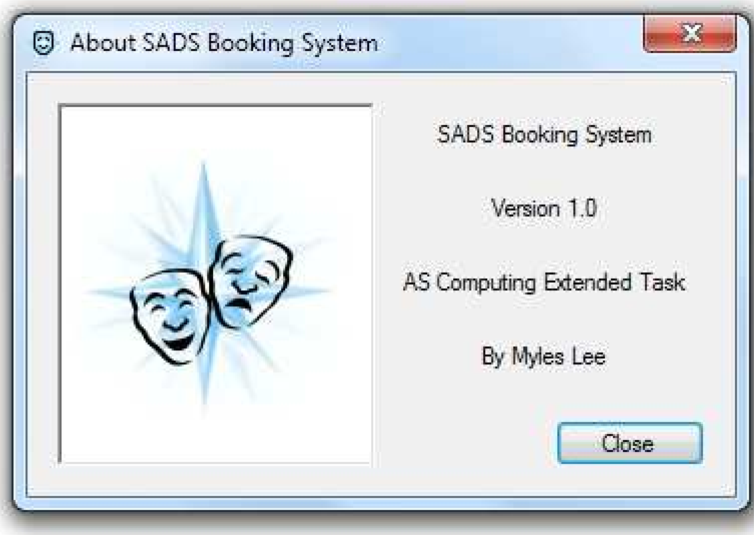
Disabled: ▨

Ticket prices	
Rows A to C	*£10.00*
Rows D to F	*£12.50*
Rows G to L	*£7.25*

To print the page, a system dialog appears, allowing the user to select the printer and appropriate preferences for printing.



The About form, provides information about the software itself.



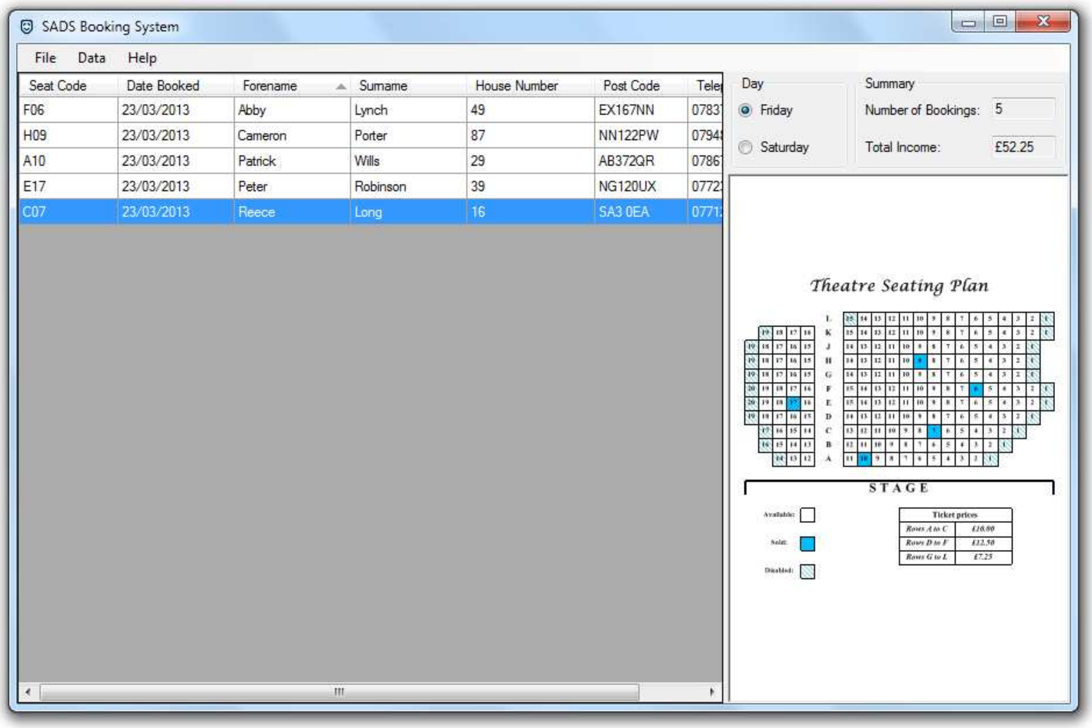
SADS Booking System

Version 1.0

AS Computing Extended Task

By Myles Lee

Records can be sorted, through the menu, for example by Forename.



Seat Code	Date Booked	Forename	Surname	House Number	Post Code	Tele
F06	23/03/2013	Abby	Lynch	49	EX167NN	0783
H09	23/03/2013	Cameron	Porter	87	NN122PW	0794
A10	23/03/2013	Patrick	Wills	29	AB372QR	0786
E17	23/03/2013	Peter	Robinson	39	NG120UX	0772
C07	23/03/2013	Reece	Long	16	SA3 0EA	0771

Day: Friday [✓]

Number of Bookings: 5

Total Income: £52.25

Ticket prices	
Rows A to C	£10.00
Rows D to F	£12.50
Rows G to L	£7.25

Annotated Listing

Login Form

```
#include "frmMain.h"

namespace SADSBookingSystem {

	using namespace System;
	using namespace System::Windows::Forms;

	public ref class frmLogin : public System::Windows::Forms::Form
	{
	public:
		frmLogin(void)
		{
			InitializeComponent();
		}
	protected:
		~frmLogin()
		{
			if (components)
			{
				delete components;
			}
		}
	private:
		System::Windows::Forms::Button^  bttLogin;
		System::Windows::Forms::Button^  bttExit;
		System::Windows::Forms::Label^  lblInfo;
		System::Windows::Forms::Label^  lblUsername;
		System::Windows::Forms::Label^  lblPassword;
		System::Windows::Forms::TextBox^  txtUsername;
		System::Windows::Forms::TextBox^  txtPassword;
		System::ComponentModel::Container ^components;
```

```
#pragma region Windows Form Designer generated code
		void InitializeComponent(void)
		{
			System::ComponentModel::ComponentResourceManager^  resources = (gcnew 
System::ComponentModel::ComponentResourceManager(frmLogin::typeid));
			this->bttLogin = (gcnew System::Windows::Forms::Button());
			this->bttExit = (gcnew System::Windows::Forms::Button());
			this->lblInfo = (gcnew System::Windows::Forms::Label());
			this->lblUsername = (gcnew System::Windows::Forms::Label());
			this->lblPassword = (gcnew System::Windows::Forms::Label());
			this->txtUsername = (gcnew System::Windows::Forms::TextBox());
			this->txtPassword = (gcnew System::Windows::Forms::TextBox());
			this->SuspendLayout();
			// 
			// bttLogin
			// 
			this->bttLogin->Location = System::Drawing::Point(221, 112);
			this->bttLogin->Name = L"bttLogin";
			this->bttLogin->Size = System::Drawing::Size(75, 23);
			this->bttLogin->TabIndex = 6;
			this->bttLogin->Text = L"&Login";
			this->bttLogin->UseVisualStyleBackColor = true;
			this->bttLogin->Click += gcnew System::EventHandler(this, &frmLogin::LoginClick);
			// 
			// bttExit
			// 
			this->bttExit->DialogResult = System::Windows::Forms::DialogResult::Cancel;
			this->bttExit->Location = System::Drawing::Point(140, 112);
			this->bttExit->Name = L"bttExit";
			this->bttExit->Size = System::Drawing::Size(75, 23);
			this->bttExit->TabIndex = 5;
			this->bttExit->Text = L"&Exit";
			this->bttExit->UseVisualStyleBackColor = true;
			this->bttExit->Click += gcnew System::EventHandler(this, &frmLogin::ExitClick);
			// 
```

```
// lblInfo
//
this->lblInfo->AutoSize = true;
this->lblInfo->Location = System::Drawing::Point(12, 9);
this->lblInfo->Name = L"lblInfo";
this->lblInfo->Size = System::Drawing::Size(284, 13);
this->lblInfo->TabIndex = 0;
this->lblInfo->Text = L"Please enter login details to access SADS booking system.";
//
// lblUsername
//
this->lblUsername->AutoSize = true;
this->lblUsername->Location = System::Drawing::Point(37, 42);
this->lblUsername->Name = L"lblUsername";
this->lblUsername->Size = System::Drawing::Size(58, 13);
this->lblUsername->TabIndex = 1;
this->lblUsername->Text = L"Username:";
//
// lblPassword
//
this->lblPassword->AutoSize = true;
this->lblPassword->Location = System::Drawing::Point(37, 77);
this->lblPassword->Name = L"lblPassword";
this->lblPassword->Size = System::Drawing::Size(56, 13);
this->lblPassword->TabIndex = 3;
this->lblPassword->Text = L"Password:";
//
// txtUsername
//
this->txtUsername->Location = System::Drawing::Point(101, 39);
this->txtUsername->Name = L"txtUsername";
this->txtUsername->Size = System::Drawing::Size(170, 20);
this->txtUsername->TabIndex = 2;
//
// txtPassword
//
```

```
			this->txtPassword->Location = System::Drawing::Point(101, 74);
			this->txtPassword->Name = L"txtPassword";
			this->txtPassword->Size = System::Drawing::Size(170, 20);
			this->txtPassword->TabIndex = 4;
			this->txtPassword->UseSystemPasswordChar = true;
			// 
			// frmLogin
			// 
			this->AcceptButton = this->bttLogin;
			this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
			this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
			this->CancelButton = this->bttExit;
			this->ClientSize = System::Drawing::Size(308, 147);
			this->Controls->Add(this->txtPassword);
			this->Controls->Add(this->txtUsername);
			this->Controls->Add(this->lblPassword);
			this->Controls->Add(this->lblUsername);
			this->Controls->Add(this->lblInfo);
			this->Controls->Add(this->bttExit);
			this->Controls->Add(this->bttLogin);
			this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedSingle;
			this->Icon = (cli::safe_cast<System::Drawing::Icon^  >(resources->GetObject(L"$this.Icon")));
			this->MaximizeBox = false;
			this->MinimizeBox = false;
			this->Name = L"frmLogin";
			this->Text = L"Login to SADS Booking System";
			this->ResumeLayout(false);
			this->PerformLayout();

		}
#pragma endregion
		//Occurs when the exit button is clicked
		void frmLogin::ExitClick(System::Object^, System::EventArgs^)
		{
			//Exit the application
			Application::Exit();
		}

		//Occurs when the login button is clicked
		void frmLogin::LoginClick(System::Object^, System::EventArgs^)
		{
			//Creates an object to hash strings, since the class' functions are not static
			Security::Cryptography::SHA512^ Encrypter = gcnew Security::Cryptography::SHA512Managed();

			//Checks whether the username and password are correct
			//The username is directly compared with the string				Username = "admin"
			//The password is hashed using SHA512 and compared to an hash	Password = "cake"
			if(txtUsername->Text == "admin" &&
				Convert::ToBase64String(
				Encrypter->ComputeHash(
				System::Text::Encoding::UTF8->GetBytes(
				txtPassword->Text))) ==
L"JDk6QFR2lZTdE4/t4QtU9fbNOnamf/itIcAMrCGRU99wU+KwghSNx/bVYw3KsbP9jsBaNs0WYV3nI4VOUlLZ4A==")
			{
				//Hides the login form
				this->Hide();

				//Creates a new form
				frmMain^ MainForm = gcnew frmMain();

				//Shows the main form
				MainForm->Show();
			} else {
				//The user typed in an incorrect username and password combination
				//Show an messagebox with an error icon
				MessageBox::Show(
					"The username or password is incorrect.",
					"Login Details Incorrect",
					MessageBoxButtons::OK,
					MessageBoxIcon::Error);
			}
			//Frees the memory allocated towards the encrypter
```

```
                        delete Encrypter;
                }
        };
}
```

Main Form

```
#pragma once

#include "frmDataEntry.h"
#include "frmAbout.h"
#include "frmFind.h"

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;
using namespace System::Drawing::Drawing2D;
using namespace System::IO;

namespace SADSBookingSystem {

        public ref class frmMain : public System::Windows::Forms::Form
        {
        public:
                frmMain(void)
                {
                        InitializeComponent();
                }

        protected:
                ~frmMain()
                {
                        if (components)
                        {
```

```
                    delete components;
                }
            }
        private: System::Windows::Forms::MenuStrip^  mnu;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuFile;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuPrintReport;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuPrintDiagram;
        private: System::Windows::Forms::ToolStripSeparator^  mnuSep1;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuExit;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuData;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuInsert;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuEdit;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuDelete;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuHelp;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuAbout;
        private: System::Windows::Forms::SplitContainer^  cntSplit;
        private: System::Windows::Forms::DataGridView^  dgvBookings;
        private: System::Windows::Forms::GroupBox^  grpDay;
        private: System::Windows::Forms::GroupBox^  grpSummary;
        private: System::Windows::Forms::Label^  lblIncomeInfo;
        private: System::Windows::Forms::Label^  lblBooksInfo;
        private: System::Windows::Forms::PictureBox^  pctDiagram;
        private: System::Windows::Forms::RadioButton^  rdbFriday;
        private: System::Windows::Forms::RadioButton^  rdbSaturday;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  SeatCode;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  DateBooked;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  Forename;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  Surname;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  HouseNumber;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  PostCode;
        private: System::Windows::Forms::DataGridViewTextBoxColumn^  Telephone;
        private: System::Windows::Forms::Panel^  grpControls;
        private: System::Windows::Forms::ToolStripSeparator^  mnuSep2;
        private: System::Windows::Forms::ToolStripMenuItem^  mnuFind;
        private: System::Drawing::Printing::PrintDocument^  dcmDiagram;
        private: System::Drawing::Printing::PrintDocument^  dcmReport;
```

```
	private: System::Windows::Forms::PrintDialog^  dlgPrint;
	private: System::Windows::Forms::PrintPreviewDialog^  dlgPreview;
	private: System::Windows::Forms::ToolStripMenuItem^  mnuPreviewReport;
	private: System::Windows::Forms::ToolStripSeparator^  mnuSep0;
	private: System::Windows::Forms::ToolStripMenuItem^  mnuPreviewDiagram;
	private: System::Windows::Forms::ToolStripSeparator^  mnuSep3;
	private: System::Windows::Forms::ToolStripMenuItem^  sortToolStripMenuItem;
	private: System::Windows::Forms::ToolStripMenuItem^  mnuSeatCode;
	private: System::Windows::Forms::ToolStripMenuItem^  mnuDateBooked;
	private: System::Windows::Forms::ToolStripMenuItem^  mnuForename;
	private: System::Windows::Forms::TextBox^  txtIncome;
	private: System::Windows::Forms::TextBox^  txtBookingsCount;
	private:
		System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
		void InitializeComponent(void)
		{
			System::ComponentModel::ComponentResourceManager^  resources = (gcnew 
System::ComponentModel::ComponentResourceManager(frmMain::typeid));
			this->mnu = (gcnew System::Windows::Forms::MenuStrip());
			this->mnuFile = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuPreviewReport = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuPrintReport = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuSep0 = (gcnew System::Windows::Forms::ToolStripSeparator());
			this->mnuPreviewDiagram = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuPrintDiagram = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuSep1 = (gcnew System::Windows::Forms::ToolStripSeparator());
			this->mnuExit = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuData = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuInsert = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuEdit = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuDelete = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuSep2 = (gcnew System::Windows::Forms::ToolStripSeparator());
			this->mnuFind = (gcnew System::Windows::Forms::ToolStripMenuItem());
			this->mnuSep3 = (gcnew System::Windows::Forms::ToolStripSeparator());
```

```
this->sortToolStripMenuItem = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->mnuSeatCode = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->mnuDateBooked = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->mnuForename = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->mnuHelp = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->mnuAbout = (gcnew System::Windows::Forms::ToolStripMenuItem());
this->cntSplit = (gcnew System::Windows::Forms::SplitContainer());
this->dgvBookings = (gcnew System::Windows::Forms::DataGridView());
this->SeatCode = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
this->DateBooked = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
this->Forename = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
this->Surname = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
this->HouseNumber = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
this->PostCode = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
this->Telephone = (gcnew System::Windows::Forms::DataGridViewTextBoxColumn());
this->pctDiagram = (gcnew System::Windows::Forms::PictureBox());
this->grpControls = (gcnew System::Windows::Forms::Panel());
this->grpDay = (gcnew System::Windows::Forms::GroupBox());
this->rdbFriday = (gcnew System::Windows::Forms::RadioButton());
this->rdbSaturday = (gcnew System::Windows::Forms::RadioButton());
this->grpSummary = (gcnew System::Windows::Forms::GroupBox());
this->txtIncome = (gcnew System::Windows::Forms::TextBox());
this->txtBookingsCount = (gcnew System::Windows::Forms::TextBox());
this->lblIncomeInfo = (gcnew System::Windows::Forms::Label());
this->lblBooksInfo = (gcnew System::Windows::Forms::Label());
this->dcmDiagram = (gcnew System::Drawing::Printing::PrintDocument());
this->dcmReport = (gcnew System::Drawing::Printing::PrintDocument());
this->dlgPrint = (gcnew System::Windows::Forms::PrintDialog());
this->dlgPreview = (gcnew System::Windows::Forms::PrintPreviewDialog());
this->mnu->SuspendLayout();
this->cntSplit->Panel1->SuspendLayout();
this->cntSplit->Panel2->SuspendLayout();
this->cntSplit->SuspendLayout();
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->dgvBookings))->BeginInit();
(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->pctDiagram))->BeginInit();
this->grpControls->SuspendLayout();
```

```
                  this->grpDay->SuspendLayout();
                  this->grpSummary->SuspendLayout();
                  this->SuspendLayout();
                  //
                  // mnu
                  //
                  this->mnu->Items->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^  >(3) {this-
>mnuFile, this->mnuData,
                        this->mnuHelp});
                  this->mnu->Location = System::Drawing::Point(0, 0);
                  this->mnu->Name = L"mnu";
                  this->mnu->Size = System::Drawing::Size(624, 24);
                  this->mnu->TabIndex = 0;
                  this->mnu->Text = L"menuStrip1";
                  //
                  // mnuFile
                  //
                  this->mnuFile->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^
>(7) {this->mnuPreviewReport,
                        this->mnuPrintReport, this->mnuSep0, this->mnuPreviewDiagram, this->mnuPrintDiagram, this-
>mnuSep1, this->mnuExit});
                  this->mnuFile->Name = L"mnuFile";
                  this->mnuFile->Size = System::Drawing::Size(37, 20);
                  this->mnuFile->Text = L"&File";
                  //
                  // mnuPreviewReport
                  //
                  this->mnuPreviewReport->Name = L"mnuPreviewReport";
                  this->mnuPreviewReport->Size = System::Drawing::Size(229, 22);
                  this->mnuPreviewReport->Text = L"Print &Preview Report...";
                  this->mnuPreviewReport->Click += gcnew System::EventHandler(this, &frmMain::PrintPreviewReport);
                  //
                  // mnuPrintReport
                  //
                  this->mnuPrintReport->Name = L"mnuPrintReport";
```

```
                    this->mnuPrintReport->ShortcutKeys = 
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control | 
System::Windows::Forms::Keys::P));
                    this->mnuPrintReport->Size = System::Drawing::Size(229, 22);
                    this->mnuPrintReport->Text = L"Print &Report...";
                    this->mnuPrintReport->Click += gcnew System::EventHandler(this, &frmMain::PrintReportClick);
                    // 
                    // mnuSep0
                    // 
                    this->mnuSep0->Name = L"mnuSep0";
                    this->mnuSep0->Size = System::Drawing::Size(226, 6);
                    // 
                    // mnuPreviewDiagram
                    // 
                    this->mnuPreviewDiagram->Name = L"mnuPreviewDiagram";
                    this->mnuPreviewDiagram->Size = System::Drawing::Size(229, 22);
                    this->mnuPreviewDiagram->Text = L"Print Pre&view Diagram...";
                    this->mnuPreviewDiagram->Click += gcnew System::EventHandler(this, &frmMain::PrintPreviewDiagram);
                    // 
                    // mnuPrintDiagram
                    // 
                    this->mnuPrintDiagram->Name = L"mnuPrintDiagram";
                    this->mnuPrintDiagram->ShortcutKeys = 
static_cast<System::Windows::Forms::Keys>(((System::Windows::Forms::Keys::Control | 
System::Windows::Forms::Keys::Shift)
                          | System::Windows::Forms::Keys::P));
                    this->mnuPrintDiagram->Size = System::Drawing::Size(229, 22);
                    this->mnuPrintDiagram->Text = L"Print &Diagram...";
                    this->mnuPrintDiagram->Click += gcnew System::EventHandler(this, &frmMain::PrintDiagramClick);
                    // 
                    // mnuSep1
                    // 
                    this->mnuSep1->Name = L"mnuSep1";
                    this->mnuSep1->Size = System::Drawing::Size(226, 6);
                    // 
                    // mnuExit
```

```
                  //
                  this->mnuExit->Name = L"mnuExit";
                  this->mnuExit->Size = System::Drawing::Size(229, 22);
                  this->mnuExit->Text = L"E&xit";
                  this->mnuExit->Click += gcnew System::EventHandler(this, &frmMain::Exit);
                  //
                  // mnuData
                  //
                  this->mnuData->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^
>(7) {this->mnuInsert, this->mnuEdit,
                        this->mnuDelete, this->mnuSep2, this->mnuFind, this->mnuSep3, this->sortToolStripMenuItem});
                  this->mnuData->Name = L"mnuData";
                  this->mnuData->Size = System::Drawing::Size(43, 20);
                  this->mnuData->Text = L"&Data";
                  //
                  // mnuInsert
                  //
                  this->mnuInsert->Name = L"mnuInsert";
                  this->mnuInsert->ShortcutKeys = System::Windows::Forms::Keys::Insert;
                  this->mnuInsert->Size = System::Drawing::Size(146, 22);
                  this->mnuInsert->Text = L"&Insert...";
                  this->mnuInsert->Click += gcnew System::EventHandler(this, &frmMain::Insert);
                  //
                  // mnuEdit
                  //
                  this->mnuEdit->Name = L"mnuEdit";
                  this->mnuEdit->ShortcutKeys =
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control |
System::Windows::Forms::Keys::E));
                  this->mnuEdit->Size = System::Drawing::Size(146, 22);
                  this->mnuEdit->Text = L"&Edit...";
                  this->mnuEdit->Click += gcnew System::EventHandler(this, &frmMain::Edit);
                  //
                  // mnuDelete
                  //
                  this->mnuDelete->Name = L"mnuDelete";
```

```
			this->mnuDelete->ShortcutKeys = System::Windows::Forms::Keys::Delete;
			this->mnuDelete->Size = System::Drawing::Size(146, 22);
			this->mnuDelete->Text = L"&Delete";
			this->mnuDelete->Click += gcnew System::EventHandler(this, &frmMain::Delete);
			// 
			// mnuSep2
			// 
			this->mnuSep2->Name = L"mnuSep2";
			this->mnuSep2->Size = System::Drawing::Size(143, 6);
			// 
			// mnuFind
			// 
			this->mnuFind->Name = L"mnuFind";
			this->mnuFind->ShortcutKeys = 
static_cast<System::Windows::Forms::Keys>((System::Windows::Forms::Keys::Control | 
System::Windows::Forms::Keys::F));
			this->mnuFind->Size = System::Drawing::Size(146, 22);
			this->mnuFind->Text = L"&Find...";
			this->mnuFind->Click += gcnew System::EventHandler(this, &frmMain::Find);
			// 
			// mnuSep3
			// 
			this->mnuSep3->Name = L"mnuSep3";
			this->mnuSep3->Size = System::Drawing::Size(143, 6);
			// 
			// sortToolStripMenuItem
			// 
			this->sortToolStripMenuItem->DropDownItems->AddRange(gcnew cli::array< 
System::Windows::Forms::ToolStripItem^  >(3) {this->mnuSeatCode, 
				this->mnuDateBooked, this->mnuForename});
			this->sortToolStripMenuItem->Name = L"sortToolStripMenuItem";
			this->sortToolStripMenuItem->Size = System::Drawing::Size(146, 22);
			this->sortToolStripMenuItem->Text = L"&Sort By ...";
			// 
			// mnuSeatCode
			// 
			this->mnuSeatCode->Name = L"mnuSeatCode";
			this->mnuSeatCode->Size = System::Drawing::Size(141, 22);
			this->mnuSeatCode->Text = L"&Seat Code";
			this->mnuSeatCode->Click += gcnew System::EventHandler(this, &frmMain::SortBySeatCode);
			// 
			// mnuDateBooked
			// 
			this->mnuDateBooked->Name = L"mnuDateBooked";
			this->mnuDateBooked->Size = System::Drawing::Size(141, 22);
			this->mnuDateBooked->Text = L"&Date Booked";
			this->mnuDateBooked->Click += gcnew System::EventHandler(this, &frmMain::SortByDateBooked);
			// 
			// mnuForename
			// 
			this->mnuForename->Name = L"mnuForename";
			this->mnuForename->Size = System::Drawing::Size(141, 22);
			this->mnuForename->Text = L"&Forename";
			this->mnuForename->Click += gcnew System::EventHandler(this, &frmMain::SortByForename);
			// 
			// mnuHelp
			// 
			this->mnuHelp->DropDownItems->AddRange(gcnew cli::array< System::Windows::Forms::ToolStripItem^ 
>(1) {this->mnuAbout});
			this->mnuHelp->Name = L"mnuHelp";
			this->mnuHelp->Size = System::Drawing::Size(44, 20);
			this->mnuHelp->Text = L"&Help";
			// 
			// mnuAbout
			// 
			this->mnuAbout->Name = L"mnuAbout";
			this->mnuAbout->ShortcutKeys = System::Windows::Forms::Keys::F1;
			this->mnuAbout->Size = System::Drawing::Size(135, 22);
			this->mnuAbout->Text = L"&About...";
			this->mnuAbout->Click += gcnew System::EventHandler(this, &frmMain::About);
			// 
			// cntSplit
			//
			this->cntSplit->Dock = System::Windows::Forms::DockStyle::Fill;
			this->cntSplit->Location = System::Drawing::Point(0, 24);
			this->cntSplit->Name = L"cntSplit";
			//
			// cntSplit.Panel1
			//
			this->cntSplit->Panel1->Controls->Add(this->dgvBookings);
			//
			// cntSplit.Panel2
			//
			this->cntSplit->Panel2->Controls->Add(this->pctDiagram);
			this->cntSplit->Panel2->Controls->Add(this->grpControls);
			this->cntSplit->Size = System::Drawing::Size(624, 418);
			this->cntSplit->SplitterDistance = 326;
			this->cntSplit->TabIndex = 0;
			//
			// dgvBookings
			//
			this->dgvBookings->AllowUserToAddRows = false;
			this->dgvBookings->AllowUserToDeleteRows = false;
			this->dgvBookings->AutoSizeColumnsMode = 
System::Windows::Forms::DataGridViewAutoSizeColumnsMode::AllCells;
			this->dgvBookings->ClipboardCopyMode = 
System::Windows::Forms::DataGridViewClipboardCopyMode::Disable;
			this->dgvBookings->ColumnHeadersHeightSizeMode = 
System::Windows::Forms::DataGridViewColumnHeadersHeightSizeMode::AutoSize;
			this->dgvBookings->Columns->AddRange(gcnew cli::array< System::Windows::Forms::DataGridViewColumn^ 
>(7) {this->SeatCode, 
				this->DateBooked, this->Forename, this->Surname, this->HouseNumber, this->PostCode, this-
>Telephone});
			this->dgvBookings->Dock = System::Windows::Forms::DockStyle::Fill;
			this->dgvBookings->EditMode = System::Windows::Forms::DataGridViewEditMode::EditProgrammatically;
			this->dgvBookings->Location = System::Drawing::Point(0, 0);
			this->dgvBookings->MultiSelect = false;
			this->dgvBookings->Name = L"dgvBookings";
			this->dgvBookings->ReadOnly = true;
			this->dgvBookings->RowHeadersVisible = false;
			this->dgvBookings->RowHeadersWidthSizeMode = 
System::Windows::Forms::DataGridViewRowHeadersWidthSizeMode::AutoSizeToDisplayedHeaders;
			this->dgvBookings->RowTemplate->Resizable = System::Windows::Forms::DataGridViewTriState::False;
			this->dgvBookings->SelectionMode = 
System::Windows::Forms::DataGridViewSelectionMode::FullRowSelect;
			this->dgvBookings->ShowCellToolTips = false;
			this->dgvBookings->ShowEditingIcon = false;
			this->dgvBookings->Size = System::Drawing::Size(326, 418);
			this->dgvBookings->TabIndex = 0;
			// 
			// SeatCode
			// 
			this->SeatCode->AutoSizeMode = 
System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCellsExceptHeader;
			this->SeatCode->HeaderText = L"Seat Code";
			this->SeatCode->MaxInputLength = 3;
			this->SeatCode->MinimumWidth = 85;
			this->SeatCode->Name = L"SeatCode";
			this->SeatCode->ReadOnly = true;
			this->SeatCode->Width = 85;
			// 
			// DateBooked
			// 
			this->DateBooked->AutoSizeMode = 
System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCellsExceptHeader;
			this->DateBooked->HeaderText = L"Date Booked";
			this->DateBooked->MaxInputLength = 32;
			this->DateBooked->MinimumWidth = 100;
			this->DateBooked->Name = L"DateBooked";
			this->DateBooked->ReadOnly = true;
			// 
			// Forename
			// 
```

```
                  this->Forename->AutoSizeMode = 
System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCellsExceptHeader;
                  this->Forename->HeaderText = L"Forename";
                  this->Forename->MaxInputLength = 64;
                  this->Forename->MinimumWidth = 100;
                  this->Forename->Name = L"Forename";
                  this->Forename->ReadOnly = true;
                  // 
                  // Surname
                  // 
                  this->Surname->AutoSizeMode = 
System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCellsExceptHeader;
                  this->Surname->HeaderText = L"Surname";
                  this->Surname->MaxInputLength = 64;
                  this->Surname->MinimumWidth = 100;
                  this->Surname->Name = L"Surname";
                  this->Surname->ReadOnly = true;
                  // 
                  // HouseNumber
                  // 
                  this->HouseNumber->AutoSizeMode = 
System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCellsExceptHeader;
                  this->HouseNumber->HeaderText = L"House Number";
                  this->HouseNumber->MaxInputLength = 5;
                  this->HouseNumber->MinimumWidth = 110;
                  this->HouseNumber->Name = L"HouseNumber";
                  this->HouseNumber->ReadOnly = true;
                  this->HouseNumber->Width = 110;
                  // 
                  // PostCode
                  // 
                  this->PostCode->AutoSizeMode = 
System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCellsExceptHeader;
                  this->PostCode->HeaderText = L"Post Code";
                  this->PostCode->MaxInputLength = 7;
                  this->PostCode->MinimumWidth = 80;
```

```
this->PostCode->Name = L"PostCode";
this->PostCode->ReadOnly = true;
this->PostCode->Width = 80;
//
// Telephone
//
this->Telephone->AutoSizeMode = System::Windows::Forms::DataGridViewAutoSizeColumnMode::AllCells;
this->Telephone->HeaderText = L"Telephone";
this->Telephone->MaxInputLength = 11;
this->Telephone->MinimumWidth = 90;
this->Telephone->Name = L"Telephone";
this->Telephone->ReadOnly = true;
this->Telephone->Width = 90;
//
// pctDiagram
//
this->pctDiagram->BackColor = System::Drawing::Color::White;
this->pctDiagram->BorderStyle = System::Windows::Forms::BorderStyle::Fixed3D;
this->pctDiagram->Dock = System::Windows::Forms::DockStyle::Fill;
this->pctDiagram->Location = System::Drawing::Point(0, 88);
this->pctDiagram->Name = L"pctDiagram";
this->pctDiagram->Size = System::Drawing::Size(294, 330);
this->pctDiagram->SizeMode = System::Windows::Forms::PictureBoxSizeMode::CenterImage;
this->pctDiagram->TabIndex = 2;
this->pctDiagram->TabStop = false;
this->pctDiagram->SizeChanged += gcnew System::EventHandler(this, &frmMain::DiagramSizeChanged);
//
// grpControls
//
this->grpControls->Controls->Add(this->grpDay);
this->grpControls->Controls->Add(this->grpSummary);
this->grpControls->Dock = System::Windows::Forms::DockStyle::Top;
this->grpControls->Location = System::Drawing::Point(0, 0);
this->grpControls->Name = L"grpControls";
this->grpControls->Size = System::Drawing::Size(294, 88);
this->grpControls->TabIndex = 3;
```

```
//
// grpDay
//
this->grpDay->Controls->Add(this->rdbFriday);
this->grpDay->Controls->Add(this->rdbSaturday);
this->grpDay->Location = System::Drawing::Point(3, 3);
this->grpDay->Name = L"grpDay";
this->grpDay->Size = System::Drawing::Size(100, 81);
this->grpDay->TabIndex = 1;
this->grpDay->TabStop = false;
this->grpDay->Text = L"Day";
//
// rdbFriday
//
this->rdbFriday->AutoSize = true;
this->rdbFriday->Checked = true;
this->rdbFriday->Location = System::Drawing::Point(6, 21);
this->rdbFriday->Name = L"rdbFriday";
this->rdbFriday->Size = System::Drawing::Size(53, 17);
this->rdbFriday->TabIndex = 0;
this->rdbFriday->TabStop = true;
this->rdbFriday->Text = L"&Friday";
this->rdbFriday->UseVisualStyleBackColor = true;
this->rdbFriday->CheckedChanged += gcnew System::EventHandler(this, &frmMain::DayChange);
//
// rdbSaturday
//
this->rdbSaturday->AutoSize = true;
this->rdbSaturday->Location = System::Drawing::Point(6, 53);
this->rdbSaturday->Name = L"rdbSaturday";
this->rdbSaturday->Size = System::Drawing::Size(67, 17);
this->rdbSaturday->TabIndex = 1;
this->rdbSaturday->Text = L"&Saturday";
this->rdbSaturday->UseVisualStyleBackColor = true;
//
// grpSummary
```

```
			//
			this->grpSummary->Anchor = 
static_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Top | 
System::Windows::Forms::AnchorStyles::Left) 
				| System::Windows::Forms::AnchorStyles::Right));
			this->grpSummary->Controls->Add(this->txtIncome);
			this->grpSummary->Controls->Add(this->txtBookingsCount);
			this->grpSummary->Controls->Add(this->lblIncomeInfo);
			this->grpSummary->Controls->Add(this->lblBooksInfo);
			this->grpSummary->Location = System::Drawing::Point(109, 3);
			this->grpSummary->Name = L"grpSummary";
			this->grpSummary->Size = System::Drawing::Size(182, 81);
			this->grpSummary->TabIndex = 0;
			this->grpSummary->TabStop = false;
			this->grpSummary->Text = L"Summary";
			// 
			// txtIncome
			// 
			this->txtIncome->Anchor = 
static_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Top | 
System::Windows::Forms::AnchorStyles::Left) 
				| System::Windows::Forms::AnchorStyles::Right));
			this->txtIncome->Location = System::Drawing::Point(118, 52);
			this->txtIncome->Name = L"txtIncome";
			this->txtIncome->ReadOnly = true;
			this->txtIncome->Size = System::Drawing::Size(55, 20);
			this->txtIncome->TabIndex = 3;
			// 
			// txtBookingsCount
			// 
			this->txtBookingsCount->Anchor = 
static_cast<System::Windows::Forms::AnchorStyles>(((System::Windows::Forms::AnchorStyles::Top | 
System::Windows::Forms::AnchorStyles::Left) 
				| System::Windows::Forms::AnchorStyles::Right));
			this->txtBookingsCount->Location = System::Drawing::Point(118, 19);
			this->txtBookingsCount->Name = L"txtBookingsCount";
```

```
                  this->txtBookingsCount->ReadOnly = true;
                  this->txtBookingsCount->Size = System::Drawing::Size(55, 20);
                  this->txtBookingsCount->TabIndex = 2;
                  //
                  // lblIncomeInfo
                  //
                  this->lblIncomeInfo->AutoSize = true;
                  this->lblIncomeInfo->Location = System::Drawing::Point(6, 55);
                  this->lblIncomeInfo->Name = L"lblIncomeInfo";
                  this->lblIncomeInfo->Size = System::Drawing::Size(72, 13);
                  this->lblIncomeInfo->TabIndex = 1;
                  this->lblIncomeInfo->Text = L"Total Income:";
                  //
                  // lblBooksInfo
                  //
                  this->lblBooksInfo->AutoSize = true;
                  this->lblBooksInfo->Location = System::Drawing::Point(6, 23);
                  this->lblBooksInfo->Name = L"lblBooksInfo";
                  this->lblBooksInfo->Size = System::Drawing::Size(106, 13);
                  this->lblBooksInfo->TabIndex = 0;
                  this->lblBooksInfo->Text = L"Number of Bookings:";
                  //
                  // dcmDiagram
                  //
                  this->dcmDiagram->PrintPage += gcnew System::Drawing::Printing::PrintPageEventHandler(this,
&frmMain::PrintDiagramDocument);
                  //
                  // dcmReport
                  //
                  this->dcmReport->PrintPage += gcnew System::Drawing::Printing::PrintPageEventHandler(this,
&frmMain::PrintReportDocument);
                  //
                  // dlgPrint
                  //
                  this->dlgPrint->UseEXDialog = true;
                  //
```

```
			// dlgPreview
			//
			this->dlgPreview->AutoScrollMargin = System::Drawing::Size(0, 0);
			this->dlgPreview->AutoScrollMinSize = System::Drawing::Size(0, 0);
			this->dlgPreview->ClientSize = System::Drawing::Size(400, 300);
			this->dlgPreview->Document = this->dcmReport;
			this->dlgPreview->Enabled = true;
			this->dlgPreview->Icon = (cli::safe_cast<System::Drawing::Icon^  >(resources-
>GetObject(L"dlgPreview.Icon")));
			this->dlgPreview->Name = L"printPreviewDialog1";
			this->dlgPreview->UseAntiAlias = true;
			this->dlgPreview->Visible = false;
			//
			// frmMain
			//
			this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
			this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
			this->ClientSize = System::Drawing::Size(624, 442);
			this->Controls->Add(this->cntSplit);
			this->Controls->Add(this->mnu);
			this->Icon = (cli::safe_cast<System::Drawing::Icon^  >(resources->GetObject(L"$this.Icon")));
			this->MainMenuStrip = this->mnu;
			this->MinimumSize = System::Drawing::Size(640, 480);
			this->Name = L"frmMain";
			this->Text = L"SADS Booking System";
			this->Load += gcnew System::EventHandler(this, &frmMain::FormLoad);
			this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this,
&frmMain::FormClose);
			this->mnu->ResumeLayout(false);
			this->mnu->PerformLayout();
			this->cntSplit->Panel1->ResumeLayout(false);
			this->cntSplit->Panel2->ResumeLayout(false);
			this->cntSplit->ResumeLayout(false);
			(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->dgvBookings))->EndInit();
			(cli::safe_cast<System::ComponentModel::ISupportInitialize^  >(this->pctDiagram))->EndInit();
			this->grpControls->ResumeLayout(false);
			this->grpDay->ResumeLayout(false);
			this->grpDay->PerformLayout();
			this->grpSummary->ResumeLayout(false);
			this->grpSummary->PerformLayout();
			this->ResumeLayout(false);
			this->PerformLayout();

		}
#pragma endregion
		//Stores the bitmap used to hold the diagram
		Bitmap^ Diagram;

		//Occurs when the form is loaded for the first time
		void frmMain::FormLoad(Object^, EventArgs^ )
		{
			this->cntSplit->Panel2MinSize = 294;
			LoadData(Application::StartupPath + "\\Friday.sadsdb");
			UpdateSummary();
			DrawDiagram(Math::Min(this->pctDiagram->ClientRectangle.Width, this->pctDiagram-
>ClientRectangle.Height));
			this->pctDiagram->Image = Diagram;
		}

		//Occurs when the form is closing
		void frmMain::FormClose(Object^, FormClosingEventArgs^ )
		{
			if(this->rdbFriday->Checked)
			{
				//Save the current changes
				SaveData(Application::StartupPath + "\\Friday.sadsdb");
			}
			else
			{
				SaveData(Application::StartupPath + "\\Saturday.sadsdb");
			}
			//Exits the application, including force closing the other forms
			Application::Exit();
		}

		//Occurs when the user changes the day
		void frmMain::DayChange(Object^, EventArgs^)
		{
			if(this->rdbFriday->Checked)
			{
				SaveData(Application::StartupPath + "\\Saturday.sadsdb");
				LoadData(Application::StartupPath + "\\Friday.sadsdb");
			} else {
				SaveData(Application::StartupPath + "\\Friday.sadsdb");
				LoadData(Application::StartupPath + "\\Saturday.sadsdb");
			}
			//Updates the summary
			UpdateSummary();
			//Updates the diagram
			DrawDiagram(Math::Min(this->pctDiagram->ClientRectangle.Width, this->pctDiagram-
>ClientRectangle.Height));
			this->pctDiagram->Image = Diagram;
		}
		//Occurs when the picture box's size has changed
		void frmMain::DiagramSizeChanged(Object^, EventArgs^)
		{
			//Updates the diagram
			DrawDiagram(Math::Min(this->pctDiagram->ClientRectangle.Width, this->pctDiagram-
>ClientRectangle.Height));
			this->pctDiagram->Image = Diagram;
		}
		//Occurs when the Exit button is clicked
		void frmMain::Exit(Object^, EventArgs^)
		{
			if(this->rdbFriday->Checked)
			{
				//Save the current changes
				SaveData(Application::StartupPath + "\\Friday.sadsdb");
```

```
        }
        else
        {
            SaveData(Application::StartupPath + "\\Saturday.sadsdb");
        }
        //Exits the application, including force closing the other forms
        Application::Exit();
    }
    //Occurs when the Insert button is clicked
    void frmMain::Insert(Object^, EventArgs^ )
    {
        //Stores whether to loop again
        bool Repeat = false;
        //Set the form as an insert form
        frmDataEntry::InsertForm = true;
        //Sets the form's day
        frmDataEntry::Day = this->rdbFriday->Checked ? L"Friday" : L"Saturday";
        //Creates a new form
        frmDataEntry^ DataForm = gcnew frmDataEntry();
        //Loops until a suitable seat code is used
        do
        {
            //Clear the seat code to determine whether a new seat is to be requested
            frmDataEntry::SeatCode = String::Empty;
            //Show the form
            DataForm->ShowDialog();
            //Calculate whether the seat is invalid and do not loop again if empty
            if(frmDataEntry::SeatCode != String::Empty)
            {
                Repeat = CheckSeat(frmDataEntry::SeatCode);
            }
            //Check whether the loop is repeat
            if(Repeat)
            {
                //Promopt the user that the seat has already been taken
```

```
                              MessageBox::Show("The seat " + frmDataEntry::SeatCode + " has already been booked.
Please choose a different seat.",
                                    "Seat Invalid", MessageBoxButtons::OK, MessageBoxIcon::Error);
                        }
                  } while(Repeat);
                  //Delete the form when it is closed
                  delete DataForm;
                  //Check if whether to add a seat
                  if(frmDataEntry::SeatCode != String::Empty)
                  {
                        //Creates an array containing the data to hold the record
                        array<Object^>^ Cells = {
                              frmDataEntry::SeatCode,
                              frmDataEntry::DateBooked.ToShortDateString(),
                              frmDataEntry::Forename,
                              frmDataEntry::Surname,
                              frmDataEntry::HouseNumber,
                              frmDataEntry::PostCode,
                              frmDataEntry::Telephone};
                        //Add a new row
                        this->dgvBookings->Rows->Add(Cells);
                        //Delete the array
                        delete Cells;
                        //Updates the summary
                        UpdateSummary();
                        //Updates the diagram
                        DrawDiagram(Math::Min(this->pctDiagram->ClientRectangle.Width, this->pctDiagram-
>ClientRectangle.Height));
                        this->pctDiagram->Image = Diagram;
                  }
            }
            //Occurs when the Edit button is clicked
            void frmMain::Edit(Object^, EventArgs^ )
            {
                  //Ensure that a record is selected
                  if(this->dgvBookings->RowCount == 0)
                  {
                        //Shows a messagebox promoting the user not to select the last row
                        MessageBox::Show("Please select a record with data.", "Cannot Edit", MessageBoxButtons::OK,
MessageBoxIcon::Error);
                  } else {
                        //Stores whether to loop again
                        bool Repeat = false;
                        //Set the form as an edit form
                        frmDataEntry::InsertForm = false;
                        //Sets the form's day
                        frmDataEntry::Day = this->rdbFriday->Checked ? L"Friday" : L"Saturday";
                        //Set the form's field properties
                        frmDataEntry::SeatCode = Convert::ToString(this->dgvBookings->SelectedCells[0]->Value);
                        frmDataEntry::DateBooked = Convert::ToDateTime(this->dgvBookings->SelectedCells[1]->Value);
                        frmDataEntry::Forename = Convert::ToString(this->dgvBookings->SelectedCells[2]->Value);
                        frmDataEntry::Surname = Convert::ToString(this->dgvBookings->SelectedCells[3]->Value);
                        frmDataEntry::HouseNumber = Convert::ToString(this->dgvBookings->SelectedCells[4]->Value);
                        frmDataEntry::PostCode = Convert::ToString(this->dgvBookings->SelectedCells[5]->Value);
                        frmDataEntry::Telephone = Convert::ToString(this->dgvBookings->SelectedCells[6]->Value);

                        //Creates a new form
                        frmDataEntry^ DataForm = gcnew frmDataEntry();
                        do
                        {
                              //Show the form
                              DataForm->ShowDialog();
                              //Calculate whether the seat is invalid and do not loop again if empty
                              Repeat = CheckSeat(frmDataEntry::SeatCode) &&
                                    frmDataEntry::SeatCode != String::Empty &&
                                    dgvBookings->SelectedRows[0]->Cells[0]->Value->ToString() !=
frmDataEntry::SeatCode;
                              //Check whether the loop is repeat
                              if(Repeat)
                              {
                                    //Promopt the user that the seat has already been taken
                                    MessageBox::Show("The seat " + frmDataEntry::SeatCode + " has already been
booked. Please choose a different seat.",
                                          "Seat Invalid", MessageBoxButtons::OK, MessageBoxIcon::Error);
                              }
                        } while(Repeat);
                        //Delete the form when it is closed
                        delete DataForm;
                        if(frmDataEntry::SeatCode != String::Empty)
                        {
                        //Set the current row's properties
                              this->dgvBookings->SelectedRows[0]->Cells[0]->Value = frmDataEntry::SeatCode;
                              this->dgvBookings->SelectedRows[0]->Cells[1]->Value =
frmDataEntry::DateBooked.ToShortDateString();
                              this->dgvBookings->SelectedRows[0]->Cells[2]->Value = frmDataEntry::Forename;
                              this->dgvBookings->SelectedRows[0]->Cells[3]->Value = frmDataEntry::Surname;
                              this->dgvBookings->SelectedRows[0]->Cells[4]->Value = frmDataEntry::HouseNumber;
                              this->dgvBookings->SelectedRows[0]->Cells[5]->Value = frmDataEntry::PostCode;
                              this->dgvBookings->SelectedRows[0]->Cells[6]->Value = frmDataEntry::Telephone;
                              //Updates the summary
                              UpdateSummary();
                              //Updates the diagram
                              DrawDiagram(Math::Min(this->pctDiagram->ClientRectangle.Width, this->pctDiagram-
>ClientRectangle.Height));
                              this->pctDiagram->Image = Diagram;
                        }
                  }
            }
            //Occurs when the Delete button is clicked
            void frmMain::Delete(Object^, EventArgs^ )
            {
                  //Checks that the selected row is in range
                  if(this->dgvBookings->RowCount == 0)
                  {
                        //Shows a messagebox promoting the user not to select the last row
                        MessageBox::Show("Please select a record with data.", "Cannot Delete",
MessageBoxButtons::OK, MessageBoxIcon::Error);
```

```
                  }
                  else
                  {
                        //Deletes the selected row
                        this->dgvBookings->Rows->RemoveAt(this->dgvBookings->SelectedRows[0]->Index);
                        //Updates the summary
                        UpdateSummary();
                        //Updates the diagram
                        DrawDiagram(Math::Min(this->pctDiagram->ClientRectangle.Width, this->pctDiagram-
>ClientRectangle.Height));
                        this->pctDiagram->Image = Diagram;              }
            }
            //Occurs when the Find button is clicked
            void frmMain::Find(Object^, EventArgs^ )
            {
                  //Creates a new form
                  frmFind^ FindForm = gcnew frmFind();
                  //Show the form
                  FindForm->ShowDialog();
                  //Delete the form when it is closed
                  delete FindForm;
                  //Checks whether a search value exists
                  if(frmFind::SearchValue != String::Empty)
                  {
                        //Checks if there are any records
                        if(this->dgvBookings->RowCount == 0)
                        {
                              //Promopt the user that the find cannot be performed
                              MessageBox::Show("There are no records to search.", "Cannot Find",
MessageBoxButtons::OK, MessageBoxIcon::Error);
                        } //Checks if the field is already selected
                        else
                        {
                              //Stores where to start searching
                              int StartRow = 0;
```

```
					//Checks whether the current row meets the search critera
					if(this->dgvBookings->SelectedRows[0]->Cells[frmFind::SearchCellIndex]->Value-
>ToString() == frmFind::SearchValue ||
						(this->dgvBookings->SelectedRows[0]->Cells[frmFind::SearchCellIndex]->Value-
>ToString()->Contains(frmFind::SearchValue) && !frmFind::SearchMatchWholeField))
					{
						//Start searching after the current row
						StartRow = this->dgvBookings->SelectedRows[0]->Index + 1;
					}

					//Loops through the records
					for(int i = StartRow; i < dgvBookings->RowCount; i++)
					{
						//Checks whether the value is found
						if(dgvBookings->Rows[i]->Cells[frmFind::SearchCellIndex]->Value->ToString() ==
frmFind::SearchValue ||
							(dgvBookings->Rows[i]->Cells[frmFind::SearchCellIndex]->Value->ToString()-
>Contains(frmFind::SearchValue) && !frmFind::SearchMatchWholeField))
						{
							//Select the current row
							this->dgvBookings->CurrentCell = this->dgvBookings->Rows[i]->Cells[0];
							//Don't search for more
							return;
						}
					}

					//Shows that the record cannot be found
					MessageBox::Show("No more records are found.", "Not Found", MessageBoxButtons::OK,
MessageBoxIcon::Error);
				}
			}
		}
		//Occurs when the About button is clicked
		void frmMain::About(Object^, EventArgs^ )
		{
			//Creates a new form
```

```
        frmAbout^ AboutForm = gcnew frmAbout();
        //Show the form
        AboutForm->ShowDialog();
        //Delete the form when it is closed
        delete AboutForm;
}
//Occurs when the report is required to be printed
void frmMain::PrintReportDocument(Object^, System::Drawing::Printing::PrintPageEventArgs^  e)
{
        //Stores the current page
        static int Page = 0;
        //Stores the current row or booking
        static int Row = 0;
        //Stores the number of rows that can be printed on one page
        int MaxRows = 53;
        //Stores the Y position of the headings
        float HeadingY = 20.0f;
        //Draws smooth and high quality text
        e->Graphics->TextRenderingHint = Drawing::Text::TextRenderingHint::AntiAlias;

        //Stores the font to display the bookings headings
        Drawing::Font^ Heading = gcnew Drawing::Font("Times New Roman", 12.0f, FontStyle::Bold);
        //Stores the font is display the infomation
        Drawing::Font^ Body = gcnew Drawing::Font("Arial Narrow", 12.0f, FontStyle::Bold);
        //Stores the font to display the title
        Drawing::Font^ TitleFont = gcnew Drawing::Font("Lucida Calligraphy", 24, FontStyle::Bold);

        //Stores the title text
        String^ TitleText = L"Theatre Bookings";
        //Stores the size of the title
        SizeF TitleSize = e->Graphics->MeasureString(TitleText, TitleFont);

        //Checks if the current page is the first page
        if(Page == 0)
        {
                //Reduces the number of rows since the first page has other content
```

```
                                MaxRows = 47;
                                //Increases the spacing for the heading on the first page
                                HeadingY = 140;
                                //Draws the title
                                e->Graphics->DrawString(TitleText, TitleFont, Brushes::Black, e->PageSettings->PaperSize-
>Width / 2 - TitleSize.Width / 2.0f, 20);
                                //Draws the summary information
                                e->Graphics->DrawString(
                                      "Day: " + (rdbFriday->Checked ? "Friday" : "Saturday") +
                                      "\nNumber of Bookings: " + txtBookingsCount->Text +
                                      "\nTotal Income: " + txtIncome->Text, Body, Brushes::Black, 70, 55);
                          }

                          //Draws the row headings
                          e->Graphics->DrawString("Seat Code", Heading, Brushes::Black, 70.0f, HeadingY);
                          e->Graphics->DrawString("Date Booked", Heading, Brushes::Black, 150.0f, HeadingY);
                          e->Graphics->DrawString("Forename", Heading, Brushes::Black, 250.0f, HeadingY);
                          e->Graphics->DrawString("Surname", Heading, Brushes::Black, 340.0f, HeadingY);
                          e->Graphics->DrawString("House Number", Heading, Brushes::Black, 430.0f, HeadingY);
                          e->Graphics->DrawString("Post Code", Heading, Brushes::Black, 550.0f, HeadingY);
                          e->Graphics->DrawString("Telephone ", Heading, Brushes::Black, 650.0f, HeadingY);

                          //Draw a black line to mark the start of data
                          e->Graphics->DrawLine(Pens::Black, 70.0f, HeadingY + 20.0f, 750.0f, HeadingY + 20.0f);

                          //Loops through the rows that can be drawn on this page
                          for(int i = 0; i < MaxRows; i++)
                          {
                                //Check if there is more rows that can be printed
                                if(Row < dgvBookings->RowCount)
                                {
                                      //Prints out the data
                                      e->Graphics->DrawString(dgvBookings->Rows[Row]->Cells[0]->Value->ToString(), Body,
Brushes::Black, 70, HeadingY + 20 + i * 20);
                                      e->Graphics->DrawString(dgvBookings->Rows[Row]->Cells[1]->Value->ToString(), Body,
Brushes::Black, 150, HeadingY + 20 + i * 20);
```

```
                              e->Graphics->DrawString(dgvBookings->Rows[Row]->Cells[2]->Value->ToString(), Body, 
Brushes::Black, 250 , HeadingY + 20 + i * 20);
                              e->Graphics->DrawString(dgvBookings->Rows[Row]->Cells[3]->Value->ToString(), Body, 
Brushes::Black, 340, HeadingY + 20 + i * 20);
                              e->Graphics->DrawString(dgvBookings->Rows[Row]->Cells[4]->Value->ToString(), Body, 
Brushes::Black, 430, HeadingY + 20 + i * 20);
                              e->Graphics->DrawString(dgvBookings->Rows[Row]->Cells[5]->Value->ToString(), Body, 
Brushes::Black, 550, HeadingY + 20 + i * 20);
                              e->Graphics->DrawString(dgvBookings->Rows[Row]->Cells[6]->Value->ToString(), Body, 
Brushes::Black, 650, HeadingY + 20 + i * 20);
                              //Increments the row counter
                              Row++;
                              //Draws a faint line to mark the next row
                              e->Graphics->DrawLine(Pens::LightGray,
                                    70.0f,
                                    HeadingY + 40.0f + Convert::ToSingle(i) * 20.0f,
                                    750.0f,
                                    HeadingY + 40.0f + Convert::ToSingle(i) * 20.0f);
                        } else {
                              //Reset the page number
                              Page = 0;
                              //Reset the row number
                              Row = 0;
                              //Exit the function
                              return;
                        }
                  }

                  //Checks if there is still more unprinted rows
                  if(Row < dgvBookings->RowCount)
                  {
                        //Ensures the other pages are printed
                        e->HasMorePages = true;
                        //Moves on to the next page
                        Page++;
                  } else {
                        //Resets the page number
                        Page = 0;
                        //Resets the row number
                        Row = 0;
                  }
            }
            //Occurs when the diagram is required to be printed
            void frmMain::PrintDiagramDocument(Object^, System::Drawing::Printing::PrintPageEventArgs^  e)
            {
                  DrawDiagram(Convert::ToInt32(Math::Min(e->PageSettings->PrintableArea.Width, e->PageSettings-
>PrintableArea.Height)));
                  e->Graphics->DrawImageUnscaled(Diagram, 0, 0);
            }
            //Occurs when the report is requied to be printed
            void frmMain::PrintReportClick(Object^, EventArgs^)
            {
                  dlgPrint->Document = dcmReport;
                  if(dlgPrint->ShowDialog() == Windows::Forms::DialogResult::OK)
                  {
                        dcmDiagram->PrinterSettings = dlgPrint->PrinterSettings;
                        dcmReport->Print();
                  }
            }
            //Occurs when the print diagram button is clicked
            void frmMain::PrintDiagramClick(Object^, EventArgs^)
            {
                  dlgPrint->Document = dcmDiagram;
                  if(dlgPrint->ShowDialog() == Windows::Forms::DialogResult::OK)
                  {
                        dcmDiagram->PrinterSettings = dlgPrint->PrinterSettings;
                        dcmDiagram->Print();
                  }
            }
            //Occurs when the report preview is requested
            void frmMain::PrintPreviewReport(Object^, EventArgs^)
            {
```

```
	dlgPreview->Document = dcmReport;
	dlgPreview->ShowDialog();
}
//Occurs when the diagram preview is requested
void frmMain::PrintPreviewDiagram(Object^, EventArgs^)
{
	dlgPreview->Document = dcmDiagram;
	dlgPreview->ShowDialog();
}
//Occurs when the Sort by Seat Code button is clicked
void frmMain::SortBySeatCode(Object^, EventArgs^)
{
	//Sort by the first column in ascending order
	this->dgvBookings->Sort(dgvBookings->Columns[0], ListSortDirection::Ascending);
}
//Occurs when the Sort by Date Booked button is clicked
void frmMain::SortByDateBooked(Object^, EventArgs^)
{
	//Sort by the first column in ascending order
	this->dgvBookings->Sort(dgvBookings->Columns[1], ListSortDirection::Ascending);
}
//Occurs when the Sort by Forename button is clicked
void frmMain::SortByForename(Object^, EventArgs^)
{
	//Sort by the first column in ascending order
	this->dgvBookings->Sort(dgvBookings->Columns[2], ListSortDirection::Ascending);
}
//Saves the current data as a custom database file to the specified filepath
void frmMain::SaveData(String^ FilePath)
{
	//Stores the medium to access to the file
	FileStream^ Stream;

	//Exposes functions for writing binary data
	BinaryWriter^ Writer;
	try
```

```
                {
                        //Opens or creates a new file for saving
                        Stream = gcnew FileStream(FilePath, FileMode::OpenOrCreate, FileAccess::Write);

                        //Links the BinaryWriter with the opened file stream
                        Writer = gcnew BinaryWriter(Stream);

                        //Loops through all the records
                        for(int i = 0; i < this->dgvBookings->RowCount; i++)
                        {
                                //Saves the Seat Code
                                Writer->Write(this->dgvBookings->Rows[i]->Cells[0]->Value->ToString());
                                //Saves the Booking Date - Coverts string into date and extracts raw int64 data to
save
                                Writer->Write(Convert::ToDateTime(this->dgvBookings->Rows[i]->Cells[1]->Value).Ticks);
                                //Saves the Customer's Forename
                                Writer->Write(this->dgvBookings->Rows[i]->Cells[2]->Value->ToString());
                                //Saves the Customer's Surname
                                Writer->Write(this->dgvBookings->Rows[i]->Cells[3]->Value->ToString());
                                //Saves the Customer's House Number
                                Writer->Write(this->dgvBookings->Rows[i]->Cells[4]->Value->ToString());
                                //Saves the Customer's Post Code
                                Writer->Write(this->dgvBookings->Rows[i]->Cells[5]->Value->ToString());
                                //Saves the Customer's Telephone
                                Writer->Write(this->dgvBookings->Rows[i]->Cells[6]->Value->ToString());
                        }

                        //End the data with a string terminator
                        Writer->Write(L"END");

                        //Closes the file
                        Writer->Close();
                        Stream->Close();

                } catch (...)
                {
                        //Promopt the user that the file could not be saved
                        MessageBox::Show("The file could not be saved.", "Cannot Save", MessageBoxButtons::OK, 
MessageBoxIcon::Error);
                }


                //Frees any memory
                delete Writer;
                delete Stream;
          }

          //Loads a custom database file from a specified filepath into the current data
          void frmMain::LoadData(String^ FilePath)
          {
                //Checks if the file exists
                if(!File::Exists(FilePath))
                {
                        //Exits the function
                        return;
                }

                //Stores the medium to access to the file
                FileStream^ Stream;

                //Exposes functions for reading binary data
                BinaryReader^ Reader;

                //Catches any errors the appears
                try
                {
                        //Opens or creates a new file for saving
                        Stream = gcnew FileStream(FilePath, FileMode::Open, FileAccess::Read);

                        //Links the BinaryWriter with the opened file stream
                        Reader = gcnew BinaryReader(Stream);
```

```
			//Clears the current data view of items
			this->dgvBookings->Rows->Clear();

			//Loops through all the recorded bookings
			for(int i = 0; i < 196; i++)
			{
				//Stores and reads the Seat Code
				String^ SeatCode = Reader->ReadString();
				//Check if the file contains no more records
				if(SeatCode == L"END")
				{
					//Exit the loop
					break;
				} else {
					//Add a new record
					this->dgvBookings->Rows->Add();
					//Sets the Seat Code
					this->dgvBookings->Rows[i]->Cells[0]->Value = SeatCode;
					//Reads the Booking Date - Reads the raw int64 and converts it into a suitable 
date format
					this->dgvBookings->Rows[i]->Cells[1]->Value = DateTime(Reader-
>ReadInt64()).ToShortDateString();
					//Reads the Customer's Forenmae
					this->dgvBookings->Rows[i]->Cells[2]->Value = Reader->ReadString();
					//Reads the Customer's Surname
					this->dgvBookings->Rows[i]->Cells[3]->Value = Reader->ReadString();
					//Reads the Customer's House Number
					this->dgvBookings->Rows[i]->Cells[4]->Value = Reader->ReadString();
					//Reads the Customer's Post Code
					this->dgvBookings->Rows[i]->Cells[5]->Value = Reader->ReadString();
					//Reads the Customer's Telephone
					this->dgvBookings->Rows[i]->Cells[6]->Value = Reader->ReadString();
				}
			}

			//Closes the file
```

```
                        Reader->Close();
                        Stream->Close();

                  } catch (...)
                  {
                        //Promopt the user that the file could not be saved
                        MessageBox::Show("The file could not be loaded.", "Cannot Load", MessageBoxButtons::OK, 
MessageBoxIcon::Error);
                  }

                  //Frees any memory
                  delete Reader;
                  delete Stream;
            }
            //Updates the summary in the top-right
            void frmMain::UpdateSummary()
            {
                  //Updates the total number of bookings
                  this->txtBookingsCount->Text = this->dgvBookings->RowCount.ToString();
                  //Stores the seat row
                  char SeatRow;
                  //Stores the total income
                  float TotalIncome = 0.0f;
                  //Loops through all the records
                  for(int i = 0; i < this->dgvBookings->RowCount; i++)
                  {
                        //Extract the seat row
                        SeatRow = Convert::ToByte(this->dgvBookings->Rows[i]->Cells[0]->Value->ToString()[0]);
                        //Comapres the seat row
                        if(SeatRow <= 'C')
                        {
                              TotalIncome += 10.00;
                        }
                        else if(SeatRow <= 'F')
                        {
                              TotalIncome += 12.50;
```

```
				}
				else
				{
					TotalIncome += 7.25;
				}
			}
			//Displays the total income
			this->txtIncome->Text = TotalIncome.ToString("£0.00");
		}
		//Draws a dynamic diagram of the theatre seating plan - Width is the width of the bitmap to generate
		void frmMain::DrawDiagram(int Width)
		{
			//Stores the number of rows
			const int SeatRowCount = 11;

			//Stores the width of one grid square
			float GridWidth = Width / 24.0f;

			//Stores the seat row
			char SeatRow = ' ';

			//Stores the seat number
			int SeatNumber = 0;

			//Stores the coordinates of the current grid square
			PointF CurrentGrid;

			//Deletes the previous bitmap if it exists
			delete Diagram;

			//Allocates memory for the bitmap
			Diagram = gcnew Bitmap(Width, Width);

			//Creates a graphic object to draw onto the bitmap
			Graphics^ Paint = Graphics::FromImage(Diagram);
```

```
                  //Ensure the graphics object draws high quality text
                  Paint->TextRenderingHint = Drawing::Text::TextRenderingHint::AntiAlias;

                  //Stores the cross hatching brush
                  Drawing::Drawing2D::HatchBrush^ DisabledBrush = gcnew HatchBrush(HatchStyle::DarkDownwardDiagonal,
Color::PowderBlue, Color::Transparent);

                  //Stores the font is diaplay the seat
                  Drawing::Font^ SeatFont = gcnew Drawing::Font("Times New Roman", GridWidth * 0.375f,
FontStyle::Bold);

                  //Stores the font to display the seat row
                  Drawing::Font^ SeatRowFont = gcnew Drawing::Font("Times New Roman", GridWidth * 0.425f,
FontStyle::Bold);

                  //Stores the font to display the seat row
                  Drawing::Font^ TicketFont = gcnew Drawing::Font("Times New Roman", GridWidth * 0.425f,
FontStyle::Bold + FontStyle::Italic);

                  //Stores the way the font is rendered
                  StringFormat^ SeatFormat = gcnew StringFormat();

                  SeatFormat->Alignment = StringAlignment::Center;
                  SeatFormat->LineAlignment = StringAlignment::Center;

                  //Stores the font to display the title
                  Drawing::Font^ TitleFont = gcnew Drawing::Font("Lucida Calligraphy", GridWidth * 0.8f,
FontStyle::Bold);

                  //Stores the title text
                  String^ TitleText = L"Theatre Seating Plan";

                  //Stores the size of the title
                  SizeF TitleSize = Paint->MeasureString(TitleText, TitleFont);

                  //Draws the title in the middle upper position
```

```
			Paint->DrawString(TitleText, TitleFont, Brushes::Black, GridWidth * 12.0f - TitleSize.Width / 
2.0f, GridWidth * 0.5f);

			//Loops through the Y axis
			for(int i = 0; i < SeatRowCount; i++)
			{
				//Sets data for the current row
				switch(i)
				{
				case 0:
					SeatRow = 'L';
					SeatNumber = 15;
					CurrentGrid.X = GridWidth * 6.0f;
					break;
				case 1:
					SeatRow = 'K';
					SeatNumber = 19;
					CurrentGrid.X = GridWidth * 2.0f;
					break;
				case 2:
					SeatRow = 'J';
					SeatNumber = 19;
					CurrentGrid.X = GridWidth;
					break;
				case 3:
					SeatRow = 'H';
					SeatNumber = 19;
					CurrentGrid.X = GridWidth;
					break;
				case 4:
					SeatRow = 'G';
					SeatNumber = 19;
					CurrentGrid.X = GridWidth;
					break;
				case 5:
					SeatRow = 'F';
```

```
		SeatNumber = 20;
		CurrentGrid.X = GridWidth;
		break;
	case 6:
		SeatRow = 'E';
		SeatNumber = 20;
		CurrentGrid.X = GridWidth;
		break;
	case 7:
		SeatRow = 'D';
		SeatNumber = 19;
		CurrentGrid.X = GridWidth;
		break;
	case 8:
		SeatRow = 'C';
		SeatNumber = 17;
		CurrentGrid.X = GridWidth * 2.0f;
		break;
	case 9:
		SeatRow = 'B';
		SeatNumber = 16;
		CurrentGrid.X = GridWidth * 2.0f;
		break;
	case 10:
		SeatRow = 'A';
		SeatNumber = 14;
		CurrentGrid.X = GridWidth * 3.0f;
		break;
	}

	CurrentGrid.Y = GridWidth * (i + 3);

	//Loops until seat number 1 is reached
	for(; SeatNumber > 0; SeatNumber--)
	{
		if(Math::Abs(CurrentGrid.X - GridWidth * 6.0f) < 0.1f)
```

```
					{
						Paint->DrawString(Char::ToString(SeatRow),
							SeatRowFont, Brushes::Black, RectangleF(CurrentGrid.X, CurrentGrid.Y,
GridWidth * 2.0f, GridWidth), SeatFormat);
						CurrentGrid.X += GridWidth * 2.0f;
					}

					if(CheckSeat(Char::ToString(SeatRow) + SeatNumber.ToString()))
					{
						Paint->FillRectangle(Brushes::DeepSkyBlue, CurrentGrid.X, CurrentGrid.Y,
GridWidth, GridWidth);
					}

					if(frmDataEntry::IsSeatDisabled(Char::ToString(SeatRow) + SeatNumber.ToString()))
					{
						Paint->FillRectangle(DisabledBrush, CurrentGrid.X, CurrentGrid.Y, GridWidth,
GridWidth);
					}

					Paint->DrawRectangle(Pens::Black, CurrentGrid.X, CurrentGrid.Y, GridWidth, GridWidth);
					Paint->DrawString(Convert::ToString(SeatNumber),
						SeatFont, Brushes::Black, RectangleF(CurrentGrid.X, CurrentGrid.Y, GridWidth,
GridWidth), SeatFormat);

					CurrentGrid.X += GridWidth;
				}
			}

			//Stores a pen which is twice as thick as default pens are
			Pen^ ThickLine = gcnew Pen(Color::Black, 2.0f);

			//Draw the stage
			Paint->DrawLine(ThickLine, GridWidth, GridWidth * 15.0f, GridWidth , GridWidth * 16.0f);
			Paint->DrawLine(ThickLine, GridWidth, GridWidth * 15.0f, GridWidth * 23.0f, GridWidth * 15.0f);
			Paint->DrawLine(ThickLine, GridWidth * 23.0f, GridWidth * 15.0f, GridWidth * 23.0f, GridWidth *
16.0f);
```

```
                  //Stores the text to display the stage
                  String^ StageText = "S T A G E";

                  //Stores the font to display the stage
                  Drawing::Font^ StageFont = gcnew Drawing::Font("Times New Roman", GridWidth * 0.7f,
FontStyle::Bold);

                  //Stores the size of the text
                  SizeF StageSize = Paint->MeasureString(StageText, StageFont);

                  //Draw the font
                  Paint->DrawString(StageText, StageFont, Brushes::Black, GridWidth * 12.0f - StageSize.Width /
2.0f, GridWidth * 15.0f);

                  //Draws the key
                  Paint->DrawString("Available:", SeatFont, Brushes::Black, RectangleF(GridWidth * 2.0f, GridWidth *
17.0f, GridWidth * 3.0f, GridWidth), SeatFormat);
                  Paint->DrawString("Sold:", SeatFont, Brushes::Black, RectangleF(GridWidth * 2.0f, GridWidth *
19.0f, GridWidth * 3.0f, GridWidth), SeatFormat);
                  Paint->DrawString("Disabled:", SeatFont, Brushes::Black, RectangleF(GridWidth * 2.0f, GridWidth *
21.0f, GridWidth * 3.0f, GridWidth), SeatFormat);
                  Paint->DrawRectangle(Pens::Black, GridWidth * 5.0f, GridWidth * 17.0f, GridWidth, GridWidth);
                  Paint->FillRectangle(Brushes::DeepSkyBlue, GridWidth * 5.0f, GridWidth * 19.0f, GridWidth,
GridWidth);
                  Paint->DrawRectangle(Pens::Black, GridWidth * 5.0f, GridWidth * 19.0f, GridWidth, GridWidth);
                  Paint->FillRectangle(DisabledBrush, GridWidth * 5.0f, GridWidth * 21.0f, GridWidth, GridWidth);
                  Paint->DrawRectangle(Pens::Black, GridWidth * 5.0f, GridWidth * 21.0f, GridWidth, GridWidth);

                  //Draws the ticket prices
                  Paint->DrawString("Ticket prices", SeatRowFont, Brushes::Black, RectangleF(GridWidth * 12.0f,
GridWidth * 17.0f, GridWidth * 8.0f, GridWidth), SeatFormat);
                  Paint->DrawString("Rows A to C", TicketFont, Brushes::Black, RectangleF(GridWidth * 12.0f,
GridWidth * 18.0f, GridWidth * 4.0f, GridWidth), SeatFormat);
                  Paint->DrawString("Rows D to F", TicketFont, Brushes::Black, RectangleF(GridWidth * 12.0f,
GridWidth * 19.0f, GridWidth * 4.0f, GridWidth), SeatFormat);
```

```
			Paint->DrawString("Rows G to L", TicketFont, Brushes::Black, RectangleF(GridWidth * 12.0f, 
GridWidth * 20.0f, GridWidth * 4.0f, GridWidth), SeatFormat);
			Paint->DrawString("£10.00", TicketFont, Brushes::Black, RectangleF(GridWidth * 16.0f, GridWidth * 
18.0f, GridWidth * 4.0f, GridWidth), SeatFormat);
			Paint->DrawString("£12.50", TicketFont, Brushes::Black, RectangleF(GridWidth * 16.0f, GridWidth * 
19.0f, GridWidth * 4.0f, GridWidth), SeatFormat);
			Paint->DrawString("£7.25", TicketFont, Brushes::Black, RectangleF(GridWidth * 16.0f, GridWidth * 
20.0f, GridWidth * 4.0f, GridWidth), SeatFormat);
			Paint->DrawRectangle(Pens::Black, GridWidth * 12.0f, GridWidth * 17.0f, GridWidth * 8.0f, 
GridWidth * 4.0f);
			Paint->DrawLine(Pens::Black, GridWidth * 12.0f, GridWidth * 18.0f, GridWidth * 20.0f, GridWidth * 
18.0f);
			Paint->DrawLine(Pens::Black, GridWidth * 12.0f, GridWidth * 19.0f, GridWidth * 20.0f, GridWidth * 
19.0f);
			Paint->DrawLine(Pens::Black, GridWidth * 12.0f, GridWidth * 20.0f, GridWidth * 20.0f, GridWidth * 
20.0f);
			Paint->DrawLine(Pens::Black, GridWidth * 16.0f, GridWidth * 18.0f, GridWidth * 16.0f, GridWidth * 
21.0f);

			delete DisabledBrush;
			delete SeatFont;
			delete SeatRowFont;
			delete TicketFont;
			delete SeatFormat;
			delete TitleFont;
			delete ThickLine;
			delete StageFont;
			delete Paint;
		}
		//Returns whether the particular seat has already been booked
		bool frmMain::CheckSeat(String^ SeatCode)
		{
			//Loops through all the records
			for(int i = 0; i < this->dgvBookings->RowCount; i++)
			{
				//Check if the selected seat is SeatCode
```

```
                        if(Convert::ToString(this->dgvBookings->Rows[i]->Cells[0]->Value) == SeatCode[0] + SeatCode-
>Substring(1)->PadLeft(2, '0'))
                        {
                              //The seat already exists
                              return true;
                        }
                  }
                  //The seat does not exist
                  return false;
            }
      };
}
```

Data Entry Form

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;


namespace SADSBookingSystem {

      /// <summary>
      /// Summary for frmDataEntry
      ///
      /// WARNING: If you change the name of this class, you will need to change the
      ///          'Resource File Name' property for the managed resource compiler tool
      ///          associated with all .resx files this class depends on.  Otherwise,
      ///          the designers will not be able to interact properly with localized
      ///          resources associated with this form.
      /// </summary>
      public ref class frmDataEntry : public System::Windows::Forms::Form
```

```
	{
	public:
		frmDataEntry(void)
		{
			InitializeComponent();
			//
			//TODO: Add the constructor code here
			//
		}

	protected:
		/// <summary>
		/// Clean up any resources being used.
		/// </summary>
		~frmDataEntry()
		{
			if (components)
			{
				delete components;
			}
		}

	private:
		/// <summary>
		/// Required designer variable.
		/// </summary>
		System::ComponentModel::Container ^components;
	private: System::Windows::Forms::Label^  lblSeatCode;
	private: System::Windows::Forms::Label^  lblDate;
	private: System::Windows::Forms::Label^  lblForename;
	private: System::Windows::Forms::Label^  lblHouseNumber;
	private: System::Windows::Forms::Button^  bttAccept;
	private: System::Windows::Forms::Button^  bttClose;
	private: System::Windows::Forms::Label^  lblSurname;
	private: System::Windows::Forms::Label^  lblTelephone;
	private: System::Windows::Forms::Label^  lblPostCode;
	private: System::Windows::Forms::MaskedTextBox^  txtSeatCode;
	private: System::Windows::Forms::DateTimePicker^  dtpDate;
	private: System::Windows::Forms::TextBox^  txtForename;
	private: System::Windows::Forms::TextBox^  txtSurname;
	private: System::Windows::Forms::MaskedTextBox^  txtHouseNumber;
	private: System::Windows::Forms::MaskedTextBox^  txtPostCode;
	private: System::Windows::Forms::MaskedTextBox^  txtTelephone;
	private: System::Windows::Forms::GroupBox^  grpCustomer;
	private: System::Windows::Forms::Label^  lblInfo;
	private: System::Windows::Forms::Label^  lblPriceInfo;

	private: System::Windows::Forms::Label^  lblDayInfo;

	private: System::Windows::Forms::Label^  lblDisabledInfo;
	private: System::Windows::Forms::TextBox^  txtPrice;
	private: System::Windows::Forms::TextBox^  txtDay;
	private: System::Windows::Forms::TextBox^  txtDisabledSeat;




	protected:


#pragma region Windows Form Designer generated code
		/// <summary>
		/// Required method for Designer support - do not modify
		/// the contents of this method with the code editor.
		/// </summary>
		void InitializeComponent(void)
		{
			this->lblSeatCode = (gcnew System::Windows::Forms::Label());
			this->lblDate = (gcnew System::Windows::Forms::Label());
			this->lblForename = (gcnew System::Windows::Forms::Label());
			this->lblHouseNumber = (gcnew System::Windows::Forms::Label());
```

```
this->bttAccept = (gcnew System::Windows::Forms::Button());
this->bttClose = (gcnew System::Windows::Forms::Button());
this->lblSurname = (gcnew System::Windows::Forms::Label());
this->lblTelephone = (gcnew System::Windows::Forms::Label());
this->lblPostCode = (gcnew System::Windows::Forms::Label());
this->txtSeatCode = (gcnew System::Windows::Forms::MaskedTextBox());
this->dtpDate = (gcnew System::Windows::Forms::DateTimePicker());
this->txtForename = (gcnew System::Windows::Forms::TextBox());
this->txtSurname = (gcnew System::Windows::Forms::TextBox());
this->txtHouseNumber = (gcnew System::Windows::Forms::MaskedTextBox());
this->txtPostCode = (gcnew System::Windows::Forms::MaskedTextBox());
this->txtTelephone = (gcnew System::Windows::Forms::MaskedTextBox());
this->grpCustomer = (gcnew System::Windows::Forms::GroupBox());
this->lblInfo = (gcnew System::Windows::Forms::Label());
this->lblPriceInfo = (gcnew System::Windows::Forms::Label());
this->lblDayInfo = (gcnew System::Windows::Forms::Label());
this->lblDisabledInfo = (gcnew System::Windows::Forms::Label());
this->txtPrice = (gcnew System::Windows::Forms::TextBox());
this->txtDay = (gcnew System::Windows::Forms::TextBox());
this->txtDisabledSeat = (gcnew System::Windows::Forms::TextBox());
this->grpCustomer->SuspendLayout();
this->SuspendLayout();
//
// lblSeatCode
//
this->lblSeatCode->AutoSize = true;
this->lblSeatCode->Location = System::Drawing::Point(13, 38);
this->lblSeatCode->Name = L"lblSeatCode";
this->lblSeatCode->Size = System::Drawing::Size(60, 13);
this->lblSeatCode->TabIndex = 1;
this->lblSeatCode->Text = L"Seat Code:";
//
// lblDate
//
this->lblDate->AutoSize = true;
this->lblDate->Location = System::Drawing::Point(13, 142);
```

```
this->lblDate->Name = L"lblDate";
this->lblDate->Size = System::Drawing::Size(73, 13);
this->lblDate->TabIndex = 9;
this->lblDate->Text = L"Date Booked:";
//
// lblForename
//
this->lblForename->AutoSize = true;
this->lblForename->Location = System::Drawing::Point(6, 22);
this->lblForename->Name = L"lblForename";
this->lblForename->Size = System::Drawing::Size(57, 13);
this->lblForename->TabIndex = 0;
this->lblForename->Text = L"Forename:";
//
// lblHouseNumber
//
this->lblHouseNumber->AutoSize = true;
this->lblHouseNumber->Location = System::Drawing::Point(6, 74);
this->lblHouseNumber->Name = L"lblHouseNumber";
this->lblHouseNumber->Size = System::Drawing::Size(81, 13);
this->lblHouseNumber->TabIndex = 4;
this->lblHouseNumber->Text = L"House Number:";
//
// bttAccept
//
this->bttAccept->Location = System::Drawing::Point(190, 326);
this->bttAccept->Name = L"bttAccept";
this->bttAccept->Size = System::Drawing::Size(75, 23);
this->bttAccept->TabIndex = 12;
this->bttAccept->Text = L"&Insert";
this->bttAccept->UseVisualStyleBackColor = true;
this->bttAccept->Click += gcnew System::EventHandler(this, &frmDataEntry::AcceptClick);
//
// bttClose
//
this->bttClose->DialogResult = System::Windows::Forms::DialogResult::Cancel;
```

```
this->bttClose->Location = System::Drawing::Point(109, 326);
this->bttClose->Name = L"bttClose";
this->bttClose->Size = System::Drawing::Size(75, 23);
this->bttClose->TabIndex = 13;
this->bttClose->Text = L"&Close";
this->bttClose->UseVisualStyleBackColor = true;
//
// lblSurname
//
this->lblSurname->AutoSize = true;
this->lblSurname->Location = System::Drawing::Point(6, 48);
this->lblSurname->Name = L"lblSurname";
this->lblSurname->Size = System::Drawing::Size(52, 13);
this->lblSurname->TabIndex = 2;
this->lblSurname->Text = L"Surname:";
//
// lblTelephone
//
this->lblTelephone->AutoSize = true;
this->lblTelephone->Location = System::Drawing::Point(6, 126);
this->lblTelephone->Name = L"lblTelephone";
this->lblTelephone->Size = System::Drawing::Size(61, 13);
this->lblTelephone->TabIndex = 8;
this->lblTelephone->Text = L"Telephone:";
//
// lblPostCode
//
this->lblPostCode->AutoSize = true;
this->lblPostCode->Location = System::Drawing::Point(6, 100);
this->lblPostCode->Name = L"lblPostCode";
this->lblPostCode->Size = System::Drawing::Size(59, 13);
this->lblPostCode->TabIndex = 6;
this->lblPostCode->Text = L"Post Code:";
//
// txtSeatCode
//
```

```
                    this->txtSeatCode->AsciiOnly = true;
                    this->txtSeatCode->Location = System::Drawing::Point(109, 35);
                    this->txtSeatCode->Mask = L">L09";
                    this->txtSeatCode->Name = L"txtSeatCode";
                    this->txtSeatCode->ResetOnSpace = false;
                    this->txtSeatCode->Size = System::Drawing::Size(150, 20);
                    this->txtSeatCode->TabIndex = 2;
                    this->txtSeatCode->TextMaskFormat = System::Windows::Forms::MaskFormat::ExcludePromptAndLiterals;
                    this->txtSeatCode->TextChanged += gcnew System::EventHandler(this, 
&frmDataEntry::SeatCodeChanged);
                    // 
                    // dtpDate
                    // 
                    this->dtpDate->Location = System::Drawing::Point(109, 139);
                    this->dtpDate->Name = L"dtpDate";
                    this->dtpDate->Size = System::Drawing::Size(150, 20);
                    this->dtpDate->TabIndex = 10;
                    // 
                    // txtForename
                    // 
                    this->txtForename->Location = System::Drawing::Point(93, 19);
                    this->txtForename->MaxLength = 32;
                    this->txtForename->Name = L"txtForename";
                    this->txtForename->Size = System::Drawing::Size(150, 20);
                    this->txtForename->TabIndex = 1;
                    // 
                    // txtSurname
                    // 
                    this->txtSurname->Location = System::Drawing::Point(93, 45);
                    this->txtSurname->MaxLength = 32;
                    this->txtSurname->Name = L"txtSurname";
                    this->txtSurname->Size = System::Drawing::Size(150, 20);
                    this->txtSurname->TabIndex = 3;
                    // 
                    // txtHouseNumber
                    // 
```

```
                  this->txtHouseNumber->AsciiOnly = true;
                  this->txtHouseNumber->Location = System::Drawing::Point(93, 71);
                  this->txtHouseNumber->Mask = L"0>CCCC";
                  this->txtHouseNumber->Name = L"txtHouseNumber";
                  this->txtHouseNumber->Size = System::Drawing::Size(150, 20);
                  this->txtHouseNumber->TabIndex = 5;
                  this->txtHouseNumber->TextMaskFormat =
System::Windows::Forms::MaskFormat::ExcludePromptAndLiterals;
                  //
                  // txtPostCode
                  //
                  this->txtPostCode->AsciiOnly = true;
                  this->txtPostCode->Location = System::Drawing::Point(93, 97);
                  this->txtPostCode->Mask = L">L\?09 0LL";
                  this->txtPostCode->Name = L"txtPostCode";
                  this->txtPostCode->Size = System::Drawing::Size(150, 20);
                  this->txtPostCode->TabIndex = 7;
                  this->txtPostCode->TextMaskFormat = System::Windows::Forms::MaskFormat::ExcludePromptAndLiterals;
                  //
                  // txtTelephone
                  //
                  this->txtTelephone->AsciiOnly = true;
                  this->txtTelephone->Location = System::Drawing::Point(93, 123);
                  this->txtTelephone->Mask = L"(9999) 000-0000";
                  this->txtTelephone->Name = L"txtTelephone";
                  this->txtTelephone->Size = System::Drawing::Size(150, 20);
                  this->txtTelephone->TabIndex = 9;
                  this->txtTelephone->TextMaskFormat = System::Windows::Forms::MaskFormat::ExcludePromptAndLiterals;
                  //
                  // grpCustomer
                  //
                  this->grpCustomer->Controls->Add(this->lblForename);
                  this->grpCustomer->Controls->Add(this->txtTelephone);
                  this->grpCustomer->Controls->Add(this->lblHouseNumber);
                  this->grpCustomer->Controls->Add(this->txtPostCode);
                  this->grpCustomer->Controls->Add(this->lblPostCode);
```

```
this->grpCustomer->Controls->Add(this->txtHouseNumber);
this->grpCustomer->Controls->Add(this->lblTelephone);
this->grpCustomer->Controls->Add(this->txtSurname);
this->grpCustomer->Controls->Add(this->lblSurname);
this->grpCustomer->Controls->Add(this->txtForename);
this->grpCustomer->Location = System::Drawing::Point(16, 165);
this->grpCustomer->Name = L"grpCustomer";
this->grpCustomer->Size = System::Drawing::Size(249, 155);
this->grpCustomer->TabIndex = 11;
this->grpCustomer->TabStop = false;
this->grpCustomer->Text = L"Customer Details";
//
// lblInfo
//
this->lblInfo->AutoSize = true;
this->lblInfo->Location = System::Drawing::Point(12, 9);
this->lblInfo->Name = L"lblInfo";
this->lblInfo->Size = System::Drawing::Size(224, 13);
this->lblInfo->TabIndex = 0;
this->lblInfo->Text = L"Please ensure that data entered are accurate.\r\n";
//
// lblPriceInfo
//
this->lblPriceInfo->AutoSize = true;
this->lblPriceInfo->Location = System::Drawing::Point(13, 64);
this->lblPriceInfo->Name = L"lblPriceInfo";
this->lblPriceInfo->Size = System::Drawing::Size(34, 13);
this->lblPriceInfo->TabIndex = 3;
this->lblPriceInfo->Text = L"Price:";
//
// lblDayInfo
//
this->lblDayInfo->AutoSize = true;
this->lblDayInfo->Location = System::Drawing::Point(13, 91);
this->lblDayInfo->Name = L"lblDayInfo";
this->lblDayInfo->Size = System::Drawing::Size(29, 13);
```

```
this->lblDayInfo->TabIndex = 5;
this->lblDayInfo->Text = L"Day:";
//
// lblDisabledInfo
//
this->lblDisabledInfo->AutoSize = true;
this->lblDisabledInfo->Location = System::Drawing::Point(13, 117);
this->lblDisabledInfo->Name = L"lblDisabledInfo";
this->lblDisabledInfo->Size = System::Drawing::Size(76, 13);
this->lblDisabledInfo->TabIndex = 7;
this->lblDisabledInfo->Text = L"Disabled Seat:";
//
// txtPrice
//
this->txtPrice->Location = System::Drawing::Point(109, 61);
this->txtPrice->Name = L"txtPrice";
this->txtPrice->ReadOnly = true;
this->txtPrice->Size = System::Drawing::Size(150, 20);
this->txtPrice->TabIndex = 4;
this->txtPrice->Text = L"Unknown";
//
// txtDay
//
this->txtDay->Location = System::Drawing::Point(109, 87);
this->txtDay->Name = L"txtDay";
this->txtDay->ReadOnly = true;
this->txtDay->Size = System::Drawing::Size(150, 20);
this->txtDay->TabIndex = 6;
this->txtDay->Text = L"Friday";
//
// txtDisabledSeat
//
this->txtDisabledSeat->Location = System::Drawing::Point(109, 113);
this->txtDisabledSeat->Name = L"txtDisabledSeat";
this->txtDisabledSeat->ReadOnly = true;
this->txtDisabledSeat->Size = System::Drawing::Size(150, 20);
```

```
this->txtDisabledSeat->TabIndex = 8;
this->txtDisabledSeat->Text = L"No";
//
// frmDataEntry
//
this->AcceptButton = this->bttAccept;
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->CancelButton = this->bttClose;
this->ClientSize = System::Drawing::Size(277, 361);
this->Controls->Add(this->txtDisabledSeat);
this->Controls->Add(this->txtDay);
this->Controls->Add(this->txtPrice);
this->Controls->Add(this->lblDisabledInfo);
this->Controls->Add(this->lblDayInfo);
this->Controls->Add(this->lblPriceInfo);
this->Controls->Add(this->lblInfo);
this->Controls->Add(this->grpCustomer);
this->Controls->Add(this->dtpDate);
this->Controls->Add(this->txtSeatCode);
this->Controls->Add(this->bttClose);
this->Controls->Add(this->bttAccept);
this->Controls->Add(this->lblDate);
this->Controls->Add(this->lblSeatCode);
this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;
this->MaximizeBox = false;
this->MinimizeBox = false;
this->Name = L"frmDataEntry";
this->ShowIcon = false;
this->Text = L"Insert Record";
this->Load += gcnew System::EventHandler(this, &frmDataEntry::FormLoad);
this->grpCustomer->ResumeLayout(false);
this->grpCustomer->PerformLayout();
this->ResumeLayout(false);
this->PerformLayout();
```

```
				}
#pragma endregion
	public:
		//Stores whether the form is used as an insert or edit form
		static bool InsertForm = true;
		//Stores the Seat Code
		static String^ SeatCode;
		//Stores the Date Booked
		static DateTime DateBooked;
		//Stores the Customer's Forename
		static String^ Forename;
		//Stores the Customer's Surname
		static String^ Surname;
		//Stores the Customer's House Number
		static String^ HouseNumber;
		//Stores the Customer's Post Code
		static String^ PostCode;
		//Stores the Customer's Telephone
		static String^ Telephone;
		//Stores the current day
		static String^ Day;

	private:
		//Returns whether all the characters in a string are not numeric
		bool frmDataEntry::IsText(String^ Value)
		{
			//Loops through all the characters
			for(int i = 0; i < Value->Length; i++)
			{
				//Checks the particular character
				if(Value[i] >= '0' && Value[i] <= '9')
				{
					//String contains numbers
					return false;
				}
			}
```

```
        //String has no numbers
        return true;
}

//Occurs when the form is started
void frmDataEntry::FormLoad(Object^, EventArgs^)
{
        //Updates the day
        this->txtDay->Text = Day;

        //Checks whether the form is an edit form
        if(!InsertForm)
        {
                //Changes the appearance of the form to suit an edit form
                this->Text = L"Edit Record";
                this->lblInfo->Text = L"Please ensure changes are accurate.";
                this->bttAccept->Text = L"&Edit";

                //Fill in empty data entry fields
                this->txtSeatCode->Text = SeatCode[0] + SeatCode->Substring(1)->PadLeft(2, '0');
                this->dtpDate->Value = DateBooked;
                this->txtForename->Text = Forename;
                this->txtSurname->Text = Surname;
                this->txtHouseNumber->Text = Convert::ToString(HouseNumber);
                this->txtPostCode->Text = PostCode;
                this->txtTelephone->Text = Telephone;
        }
}

//Occurs when the Insert or Edit button is pressed
void frmDataEntry::AcceptClick(Object^, EventArgs^)
{
        //Extracts the first character, and puts a rouge value if no character is present
        char SeatRow = Convert::ToByte(txtSeatCode->Text->PadRight(1, 'Z')[0]);

        //Extracts the following chracters, and puts a rouge value if no characters are present
```

```
                    int SeatNumber = Convert::ToInt32(txtSeatCode->Text->PadRight(2, '0')->Substring(1));

                    //Performs validation of fields that has been entered and displays the messagebox to prompt the
user
                    //Also, the field that is incorrectly entered is selected
                    if(SeatNumber == 0) //Seat number is 1 or more
                    {
                          MessageBox::Show("Please enter a valid seat number.", "Invalid Data Entry",
MessageBoxButtons::OK, MessageBoxIcon::Error);
                          this->txtSeatCode->Focus();
                    }
                    else if(SeatRow < L'A' || SeatRow > L'L' || SeatRow == L'I') //Seat row is between A and L,
excluding I
                    {
                          MessageBox::Show("Please enter a valid seat row.", "Invalid Data Entry",
MessageBoxButtons::OK, MessageBoxIcon::Error);
                          this->txtSeatCode->Focus();
                    }
                    else if( SeatRow == L'A' && SeatNumber > 14 ||
                          SeatRow == L'B' && SeatNumber > 16 ||
                          SeatRow == L'C' && SeatNumber > 17 ||
                          SeatRow == L'D' && SeatNumber > 19 ||
                          SeatRow == L'E' && SeatNumber > 20 ||
                          SeatRow == L'F' && SeatNumber > 20 ||
                          SeatRow == L'G' && SeatNumber > 19 ||
                          SeatRow == L'H' && SeatNumber > 19 ||
                          SeatRow == L'J' && SeatNumber > 19 ||
                          SeatRow == L'K' && SeatNumber > 19 ||
                          SeatRow == L'L' && SeatNumber > 15) //Ensures the selected row has the right number of seats
                    {
                          MessageBox::Show("Please enter a valid seat number.", "Invalid Data Entry",
MessageBoxButtons::OK, MessageBoxIcon::Error);
                          this->txtSeatCode->Focus();
                    }
                    else if(dtpDate->Value < DateTime(1996, 1, 1) || dtpDate->Value > DateTime::Now) //Stops future
booking
```

```
				{
					MessageBox::Show("Please enter a valid booking date.", "Invalid Data Entry", 
MessageBoxButtons::OK, MessageBoxIcon::Error);
					this->dtpDate->Focus();
				}
				else if(!IsText(txtForename->Text) || txtForename->Text == String::Empty) //Stops numbers in names 
and presence check
				{
					MessageBox::Show("Please enter a valid forename.", "Invalid Data Entry", 
MessageBoxButtons::OK, MessageBoxIcon::Error);
					this->txtForename->Focus();
				}
				else if(!IsText(txtSurname->Text) || txtSurname->Text == String::Empty) //Stops numbers in names 
and presence check
				{
					MessageBox::Show("Please enter a valid surname.", "Invalid Data Entry", 
MessageBoxButtons::OK, MessageBoxIcon::Error);
					this->txtSurname->Focus();
				}
				else if(txtHouseNumber->Text == String::Empty) //House number cannot be 0
				{
					MessageBox::Show("Please enter a valid house number.", "Invalid Data Entry", 
MessageBoxButtons::OK, MessageBoxIcon::Error);
					this->txtHouseNumber->Focus();
				}
				else if(txtPostCode->Text->Length < 6) //Post code must be entered
				{
					MessageBox::Show("Post code is too short.", "Invalid Data Entry", MessageBoxButtons::OK, 
MessageBoxIcon::Error);
					this->txtPostCode->Focus();
				}
				else if(txtTelephone->Text->Length < 7) //Telephone must be entered
				{
					MessageBox::Show("Telephone number is too short.", "Invalid Data Entry", 
MessageBoxButtons::OK, MessageBoxIcon::Error);
					this->txtTelephone->Focus();
```

```
                }
                else
                {
                        //Apply the form's fields to the static variables
                        SeatCode = this->txtSeatCode->Text[0] + this->txtSeatCode->Text->Substring(1)->PadLeft(2,
'0');

                        DateBooked = this->dtpDate->Value;
                        Forename = this->txtForename->Text;
                        Surname = this->txtSurname->Text;
                        HouseNumber = this->txtHouseNumber->Text;
                        PostCode = this->txtPostCode->Text;
                        Telephone = this->txtTelephone->Text;

                        //Close the form
                        Close();
                }
        }

        //Occurs when the Seat Code is changed
        void frmDataEntry::SeatCodeChanged(Object^, EventArgs^)
        {
                //Extracts the first character, and puts a rouge value if no character is present
                char SeatRow = Convert::ToByte(txtSeatCode->Text->PadRight(1, 'Z')[0]);

                //Compares the row with the pricing
                if(SeatRow <= L'C')
                {
                        txtPrice->Text = L"£10.00";
                }
                else if(SeatRow <= L'F')
                {
                        txtPrice->Text = L"£12.50";
                }
                else if(SeatRow <= L'L' && SeatRow != 'I')
                {
                        txtPrice->Text = L"£7.25";
```

```
				}
				else
				{
					txtPrice->Text = L"Unknown";
				}

				if(IsSeatDisabled(txtSeatCode->Text))
				{
					txtDisabledSeat->Text = "Yes";
				} else {
					txtDisabledSeat->Text = "No";
				}
			}

	public:
			//Returns whether the seat is disabled
			static bool frmDataEntry::IsSeatDisabled(String^ SeatCode)
			{
				//Extracts the first character, and puts a rouge value if no character is present
				char SeatRow = Convert::ToByte(SeatCode->PadRight(1, 'Z')[0]);

				//Extracts the following chracters, and puts a rouge value if no characters are present
				int SeatNumber = Convert::ToInt32(SeatCode->PadRight(2, '0')->Substring(1));

				//Checks if the seat is the first row
				if(SeatNumber == 1)
				{
					return true;
				} else if(
						(SeatRow == 'A' && SeatNumber == 14) ||
						(SeatRow == 'B' && SeatNumber == 16) ||
						(SeatRow == 'C' && SeatNumber == 17) ||
						(SeatRow == 'D' && SeatNumber == 19) ||
						(SeatRow == 'E' && SeatNumber == 20) ||
						(SeatRow == 'F' && SeatNumber == 20) ||
						(SeatRow == 'G' && SeatNumber == 19) ||
                                                (SeatRow == 'H' && SeatNumber == 19) ||
                                                (SeatRow == 'J' && SeatNumber == 19) ||
                                                (SeatRow == 'K' && SeatNumber == 19) ||
                                                (SeatRow == 'L' && SeatNumber == 15))
                                {
                                        return true;
                                } else {
                                        return false;
                                }
                        }
                };
}
```

Find Form

```
#pragma once

using namespace System;
using namespace System::ComponentModel;
using namespace System::Collections;
using namespace System::Windows::Forms;
using namespace System::Data;
using namespace System::Drawing;


namespace SADSBookingSystem {

        /// <summary>
        /// Summary for frmFind
        ///
        /// WARNING: If you change the name of this class, you will need to change the
        ///          'Resource File Name' property for the managed resource compiler tool
        ///          associated with all .resx files this class depends on.  Otherwise,
        ///          the designers will not be able to interact properly with localized
        ///          resources associated with this form.
        /// </summary>
        public ref class frmFind : public System::Windows::Forms::Form
```

```
{
public:
      frmFind(void)
      {
            InitializeComponent();
            //
            //TODO: Add the constructor code here
            //
      }

protected:
      /// <summary>
      /// Clean up any resources being used.
      /// </summary>
      ~frmFind()
      {
            if (components)
            {
                  delete components;
            }
      }
private: System::Windows::Forms::Label^  lblFind;
private: System::Windows::Forms::TextBox^  txtFind;
private: System::Windows::Forms::RadioButton^  rdbSeatCode;
private: System::Windows::Forms::RadioButton^  rdbSurname;
private: System::Windows::Forms::RadioButton^  rdbPostCode;
private: System::Windows::Forms::Button^  bttClose;
private: System::Windows::Forms::Button^  bttFind;


protected:
      private: System::Windows::Forms::GroupBox^  grpSearchRange;
      private: System::Windows::Forms::CheckBox^  chcWholeField;


      protected:

      private:
            /// <summary>
            /// Required designer variable.
            /// </summary>
            System::ComponentModel::Container ^components;

#pragma region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            void InitializeComponent(void)
            {
                  this->lblFind = (gcnew System::Windows::Forms::Label());
                  this->txtFind = (gcnew System::Windows::Forms::TextBox());
                  this->rdbSeatCode = (gcnew System::Windows::Forms::RadioButton());
                  this->rdbSurname = (gcnew System::Windows::Forms::RadioButton());
                  this->rdbPostCode = (gcnew System::Windows::Forms::RadioButton());
                  this->bttClose = (gcnew System::Windows::Forms::Button());
                  this->bttFind = (gcnew System::Windows::Forms::Button());
                  this->grpSearchRange = (gcnew System::Windows::Forms::GroupBox());
                  this->chcWholeField = (gcnew System::Windows::Forms::CheckBox());
                  this->grpSearchRange->SuspendLayout();
                  this->SuspendLayout();
                  //
                  // lblFind
```

```
//
this->lblFind->AutoSize = true;
this->lblFind->Location = System::Drawing::Point(9, 15);
this->lblFind->Name = L"lblFind";
this->lblFind->Size = System::Drawing::Size(30, 13);
this->lblFind->TabIndex = 1;
this->lblFind->Text = L"Find:";
//
// txtFind
//
this->txtFind->Location = System::Drawing::Point(45, 12);
this->txtFind->MaxLength = 32;
this->txtFind->Name = L"txtFind";
this->txtFind->Size = System::Drawing::Size(123, 20);
this->txtFind->TabIndex = 2;
//
// rdbSeatCode
//
this->rdbSeatCode->AutoSize = true;
this->rdbSeatCode->Checked = true;
this->rdbSeatCode->Location = System::Drawing::Point(6, 19);
this->rdbSeatCode->Name = L"rdbSeatCode";
this->rdbSeatCode->Size = System::Drawing::Size(75, 17);
this->rdbSeatCode->TabIndex = 4;
this->rdbSeatCode->TabStop = true;
this->rdbSeatCode->Text = L"Seat Code";
this->rdbSeatCode->UseVisualStyleBackColor = true;
//
// rdbSurname
//
this->rdbSurname->AutoSize = true;
this->rdbSurname->Location = System::Drawing::Point(6, 42);
this->rdbSurname->Name = L"rdbSurname";
this->rdbSurname->Size = System::Drawing::Size(114, 17);
this->rdbSurname->TabIndex = 5;
this->rdbSurname->Text = L"Customer Surname";
```

```
this->rdbSurname->UseVisualStyleBackColor = true;
//
// rdbPostCode
//
this->rdbPostCode->AutoSize = true;
this->rdbPostCode->Location = System::Drawing::Point(6, 65);
this->rdbPostCode->Name = L"rdbPostCode";
this->rdbPostCode->Size = System::Drawing::Size(121, 17);
this->rdbPostCode->TabIndex = 7;
this->rdbPostCode->Text = L"Customer Post Code";
this->rdbPostCode->UseVisualStyleBackColor = true;
//
// bttClose
//
this->bttClose->DialogResult = System::Windows::Forms::DialogResult::Cancel;
this->bttClose->Location = System::Drawing::Point(12, 155);
this->bttClose->Name = L"bttClose";
this->bttClose->Size = System::Drawing::Size(75, 23);
this->bttClose->TabIndex = 8;
this->bttClose->Text = L"Close";
this->bttClose->UseVisualStyleBackColor = true;
this->bttClose->Click += gcnew System::EventHandler(this, &frmFind::CloseClick);
//
// bttFind
//
this->bttFind->Location = System::Drawing::Point(93, 155);
this->bttFind->Name = L"bttFind";
this->bttFind->Size = System::Drawing::Size(75, 23);
this->bttFind->TabIndex = 9;
this->bttFind->Text = L"Find";
this->bttFind->UseVisualStyleBackColor = true;
this->bttFind->Click += gcnew System::EventHandler(this, &frmFind::FindClick);
//
// grpSearchRange
//
this->grpSearchRange->Controls->Add(this->rdbSeatCode);
```

```
this->grpSearchRange->Controls->Add(this->rdbPostCode);
this->grpSearchRange->Controls->Add(this->rdbSurname);
this->grpSearchRange->Location = System::Drawing::Point(12, 61);
this->grpSearchRange->Name = L"grpSearchRange";
this->grpSearchRange->Size = System::Drawing::Size(156, 88);
this->grpSearchRange->TabIndex = 10;
this->grpSearchRange->TabStop = false;
this->grpSearchRange->Text = L"Search Range";
//
// chcWholeField
//
this->chcWholeField->AutoSize = true;
this->chcWholeField->Location = System::Drawing::Point(12, 38);
this->chcWholeField->Name = L"chcWholeField";
this->chcWholeField->Size = System::Drawing::Size(109, 17);
this->chcWholeField->TabIndex = 11;
this->chcWholeField->Text = L"Match whole &field";
this->chcWholeField->UseVisualStyleBackColor = true;
//
// frmFind
//
this->AcceptButton = this->bttFind;
this->AutoScaleDimensions = System::Drawing::SizeF(6, 13);
this->AutoScaleMode = System::Windows::Forms::AutoScaleMode::Font;
this->CancelButton = this->bttClose;
this->ClientSize = System::Drawing::Size(180, 187);
this->Controls->Add(this->chcWholeField);
this->Controls->Add(this->grpSearchRange);
this->Controls->Add(this->bttFind);
this->Controls->Add(this->bttClose);
this->Controls->Add(this->txtFind);
this->Controls->Add(this->lblFind);
this->FormBorderStyle = System::Windows::Forms::FormBorderStyle::FixedDialog;
this->MaximizeBox = false;
this->MinimizeBox = false;
this->Name = L"frmFind";
```

```
			this->Text = L"Find";
			this->Load += gcnew System::EventHandler(this, &frmFind::FormLoad);
			this->FormClosing += gcnew System::Windows::Forms::FormClosingEventHandler(this, 
&frmFind::FormClose);
			this->grpSearchRange->ResumeLayout(false);
			this->grpSearchRange->PerformLayout();
			this->ResumeLayout(false);
			this->PerformLayout();

		}
#pragma endregion
	public:
		//Stores the search value
		static String^ SearchValue;

		//Stores the column index to search within
		static int SearchCellIndex;

		//Stores whether to search for the whole word
		static bool SearchMatchWholeField = false;

	private:
		//Occurs when the form is started
		void frmFind::FormLoad(Object^, EventArgs^)
		{
			//Selects which radio button is to be checked
			switch(this->SearchCellIndex)
			{
			case 0:
				this->rdbSeatCode->Checked = true;
				break;
			case 3:
				this->rdbSurname->Checked = true;
				break;
			case 5:
				this->rdbPostCode->Checked = true;
				}

				//Restore settings from the form being closed
				this->txtFind->Text = SearchValue;
				this->chcWholeField->Checked = SearchMatchWholeField;
			}

			//Occurs when the Find button is clicked
			void frmFind::FindClick(Object^, EventArgs^)
			{
				//Checks if the Search Value has been entered
				if(this->txtFind->Text == String::Empty)
				{
					//Promopt the user to enter in the search value
					MessageBox::Show("Please enter in what to find.", "Missing Search Value",
MessageBoxButtons::OK, MessageBoxIcon::Error);
				} else {
					//Sets the search value
					this->SearchValue = this->txtFind->Text;

					//Selects which radio button is checked, and applied the coorsponding column index
					if(rdbSeatCode->Checked)
					{
						this->SearchCellIndex = 0;
					} else if(rdbSurname->Checked)
					{
						this->SearchCellIndex = 3;
					} else {
						this->SearchCellIndex = 5;
					}

					//Sets whether to search for a whole word
					this->SearchMatchWholeField = this->chcWholeField->Checked;

					//Closes the form
					this->Close();
```

```
                  }
            }

            //Occurs when the Close button is clicked
            void frmFind::CloseClick(Object^, EventArgs^)
            {
                  //Clears the Search Value
                  this->SearchValue = String::Empty;
            }

            //Occurs when the form is closing
            void frmFind::FormClose(System::Object^, System::Windows::Forms::FormClosingEventArgs^)
            {
                  //If the use typed in a valid search value, do not clear it
                  if(this->SearchValue != this->txtFind->Text)
                  {
                        //Clear the search value when the user did not press the close button
                        this->SearchValue = String::Empty;
                  }
            }
      };
}
```

Evaluation and Testing

Testing Strategy

Applications need to be tested thoroughly to make sure that are at the required standard of their users. Testing identifies errors and problems within the application to allow them to be fixed in future updates or patches. My Testing Strategy will compose of various sections to meet standards, which would include:

- Ways of detected errors
- Consistent and modular checking
- Syntax, or spelling errors
- Visual errors, such as using the wrong colour

Navigational Testing

Navigational testing will allow the user to access all the forms, and access all the features (whether they work or not). This is important as even if the functionality works, there it is redundant when the user cannot even access it. Additionally, Navigational Testing involves testing the clarity of the user interface, so the user can use the application effortless. This would involve utilising features offered by the .NET controls, such as the menu accessibility, using the ALT key. Overall, the Navigational Testing would use a lot menu clicking.

Functionality Testing

Functional testing is ensuring that the program functions as they way that is useful towards the end user. If the program does not do what the user expects, then needs to be corrected, because it is the primary goal of the program. Apart from the general functionality, the instructions from the user need to performed efficiently. This requires the code to be executed at a fast speed, otherwise it would give the impression that the program has crashed, and even through it is doing its job.

Data Handling Testing

The program allows the user to manipulate data about the seat bookings, so it should be able to perform tasks such as:

- Searching for different bookings in a range to criteria, using the Find dialog to locate that particular record
- Sorting bookings depending on their fields, by selected commands through the main menu
- Calculation of the revenue in both days, as displayed by the top-right corner, in the summary group box
- Data verification and validation, as shown by the data entry form including the: Insert Form and the Edit Form. The controls would be selected for the verification features, and the submit button will execute the validation of the entered data. This would be testing by using test data, which is a section below, in the functionality part.

If these were to go wrong, a fast valid accurate response is required to tell the end user what to do about it.

Hardware Compatibility Testing

The program requires its features to run efficiently, without compromising the limitations of the end user's computer. This involves maintaining the memory usage, and CPU usage to a minimum, so it works on the computer. This can be tested by using Task Manager, by monitoring the memory usage, so it can detect any memory leaks within any calculations of the program. This would be tested while other test would be carried out, so that it reflects an accurate usage of the program. The CPU usage would be tested by the response time, and Task Manager. This would indicate any infinite loops, were the CPU would be running continuously at a high percentage.

Interaction Testing

Interaction testing is making sure that the program visually looks like it does its job. For example, when a record in inserted, a new record is displayed in the Data View Grid. This works by posting a new record, or in other words, adding a new record, using a temporary array. In addition, the data would need to be displayed correctly, as if the program was to be re-opened , the Data View Grid is blank, then it would be confusing for what the user should be doing. Therefore, a consistent algorithm, with error checking is required to maintain a data that is available to the user at all times.

Test Data

There would be many different types of data to test how robust my program is. The program should be able to accept incorrect or correct data, by doing the appropriate task. There are four different types of test data:

- Normal – This data is what the program is meant to handle, and should show no problem, as it is expected. For example, Myles for forename.
- Erroneous – This data is that incorrect, but can be entered, such as "Bob1" for name, since it is a string. Validation should detect this and alert the user.
- Extreme – This data should work, but may cause problems, such as a name which has many characters.
- Invalid – This data is of the wrong data type, such as "-123456" for Post Code. My program has verification to prevent data of such type.
- None – This is where there is no data enter for the field.

Data that would cause errors is difficult to test to the use of masked textboxes, and correct usage of controls, the verification stops the user entering such data. When the tries

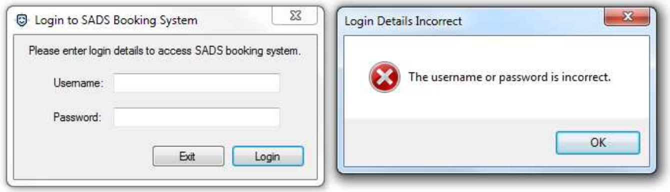
Test Runs

Testing Table

Test #	Category	Test Object	Data Value	Testing Type	Expected Outcome	Actual Outcome	Screenshot #
1	Functionality	Login Username & Password Textbox	“”	None	Message Box	Message Box	1
2	Functionality	Login Username & Password Textbox	“123456789”	Erroneous	Message Box	Message Box	2
3	Functionality	Seat Code Textbox	“”	None	Message Box	Message Box	3
4	Functionality	Seat Code Textbox	“A99”	Erroneous	Message Box	Message Box	3
5	Functionality	Seat Code Textbox	“A10”	Erroneous	Message Box	Message Box	4
6	Functionality	Forename Textbox	“123456789”	Invalid	Message Box	Message Box	5
7	Functionality	Surname Textbox	“123456789”	Invalid	Message Box	Message Box	6
8	Functionality	Forename Textbox	32 Characters of ‘A’	Extreme	New Record	New Record	7
9	Functionality	House Number Textbox	“”	None	Message Box	Message Box	8
10	Functionality	Post Code	“AA11 1AA”	Normal	New Record	New Record	9
11	Functionality	Telephone	“(____) __3-8___”	Erroneous	Message Box	New Record	10
12	Navigation	Menu	N/A	N/A	Menu Item Selection	Menu Item Selection	11
13	Data Handling	Sort by Seat Code	N/A	N/A	Items sorted by seat code.	Items sorted.	12
14	Data Handling	Sort by Date Booked	N/A	N/A	Items sorted by date booked.	Items sorted.	12
15	Data Handling	Sort by Forename	N/A	N/A	Items sorted by forename.	Items sorted.	13
16	Data Handling	Search for Seat Code	“”	None	Message Box	Message Box	14
17	Data Handling	Search for Seat Code	“123456”	Normal	Message Box	Message Box	15
18	Data Handling	Search for Surname	“Wills”	Normal	Message Box	Message Box	16
19	Data Handling	Revenue Calculation	N/A	N/A	£55.25	£52.25	17

20	Hardware Compatibility	Application	N/A	N/A	10MB	7,916KB	N/A
21	Interaction	New Record	N/A	N/A	New Record	New Record	18
22	Interaction	Edit Record	N/A	N/A	Edit Record	Edit Record	19
23	Interaction	Delete Record	N/A	N/A	Delete Record	Delete Record	20
24	Interaction	Print Preview Report	N/A	N/A	Print Preview Dialog	Print Preview Dialog	21
25	Interaction	Print Report	N/A	N/A	Print Dialog	Print Dialog	22
26	Interaction	Print Preview Diagram	N/A	N/A	Print Preview Dialog	Print Preview Dialog	23
27	Interaction	Print Diagram	N/A	N/A	Print Dialog	Print Dialog	22
28	Interaction	About Form Button	N/A	N/A	About Form	About Form	24
28	Interaction	Friday & Saturday Buttons	N/A	N/A	Switch to Saturday	Switch to Saturday	25
29	Interaction	Login Button	N/A	N/A	Goes to the Main Form	Goes to the Main Form	26
30	Interaction	Exit Button	N/A	N/A	Closes the program	Closes the program	27
31	Interaction	Delete Database	N/A	N/A	Empty Table	Empty Table	28

Testing Screenshots

Screenshot #	Comment	Screenshot Image
1	No data is entered for any text boxes, so a message box appears.	

Login to SADS Booking System

Please enter login details to access SADS booking system.

Username:

Password:

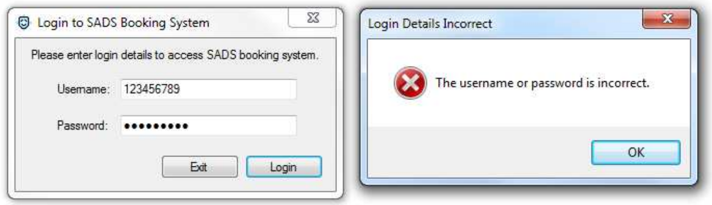
Exit Login

Login Details Incorrect

The username or password is incorrect.

OK

2 Incorrect data is entered, since the username and password are wrong. No more testing for the Login Form is required, as it works as the other forms can be tested successfully.



Login to SADS Booking System

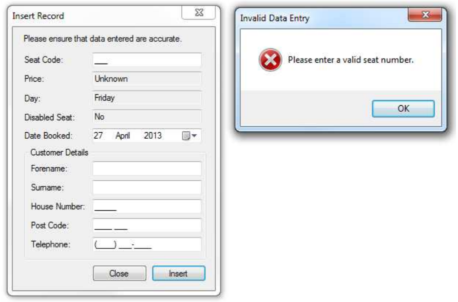
Please enter login details to access SADS booking system.

Username:	123456789
Password:	•••••••••

Login Details Incorrect

The username or password is incorrect.

3 An invalid Seat Code, regardless of the test type being None or Erroneous, shows the same message box, expect for the next screenshot.



Insert Record

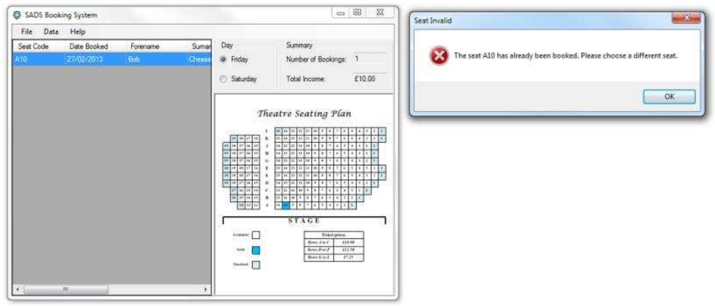
Please ensure that data entered are accurate.

Seat Code:	__
Price:	Unknown
Day:	Friday
Disabled Seat:	No
Date Booked:	27 April 2013
Customer Details	
Forename:	
Surname:	
House Number:	____
Post Code:	___ ___
Telephone:	(___) ___-____

Invalid Data Entry

Please enter a valid seat number.

4 If the seat has already been booked, another message box will appear. This shows that the user cannot double book the same seat.



SADS Booking System

Seat Code	Date Booked	Forename	Surnar
A10	27/02/2013	Bob	Cheese

Day: Friday (selected), Saturday

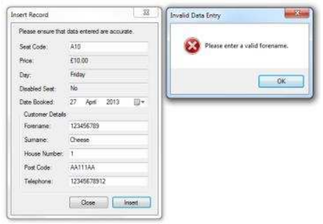
Summary

Number of Bookings:	1
Total Income:	£10.00

Seat Invalid

The seat A10 has already been booked. Please choose a different seat.

5 Even through the forename is a string, the validation check would check for invalid characters, such as numbers.



Insert Record

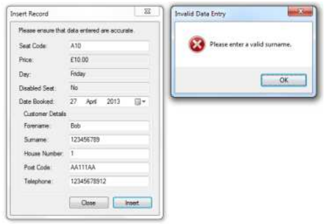
Please ensure that data entered are accurate.

Seat Code:	A10
Price:	£10.00
Day:	Friday
Disabled Seat:	No
Date Booked:	27 April 2013
Customer Details	
Forename:	123456789
Surname:	Cheese
House Number:	1
Post Code:	AA111AA
Telephone:	12345678912

Invalid Data Entry

Please enter a valid forename.

6 The surname has the identical check as the forename, as the same function is used, as demonstrated by the code. However, a different message box is shown, where forename is replaced with surname.



Insert Record

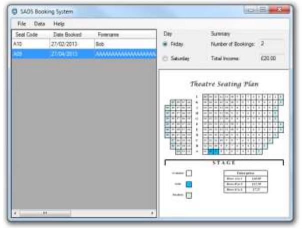
Please ensure that data entered are accurate.

Seat Code:	A10
Price:	£10.00
Day:	Friday
Disabled Seat:	No
Date Booked:	27 April 2013
Customer Details	
Forename:	Bob
Surname:	123456789
House Number:	1
Post Code:	AA111AA
Telephone:	12345678912

Invalid Data Entry

Please enter a valid surname.

7 A new record is placed, and the forename header resizes to cope with 32 characters. However, it is rare the user would do this.

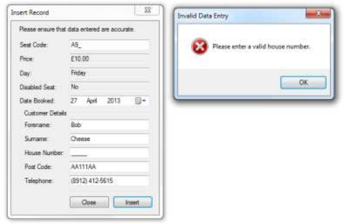


SADS Booking System

Seat Code	Date Booked	Forename
A10	27/02/2013	Bob
A09	27/04/2013	AAAAAAAAAAAAAAAAA

Day: Friday / Saturday — Summary: Number of Bookings: 2; Total Income: £20.00

8 The post code displays as separate message box.



Insert Record

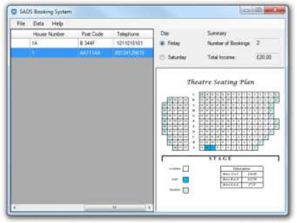
Please ensure that data entered are accurate.

Seat Code:	A5_
Price:	£10.00
Day:	Friday
Disabled Seat:	No
Date Booked:	27 April 2013
Customer Details	
Forename:	Bob
Surname:	Cheese
House Number:	____
Post Code:	AA111AA
Telephone:	(8912) 412-5615

Invalid Data Entry

Please enter a valid house number.

9 The new post code appears on the Data View Grid, without any problems.

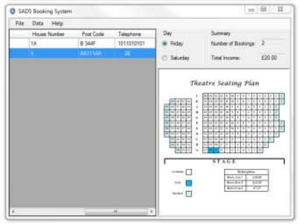


SADS Booking System

House Number	Post Code	Telephone
1A	B 344F	1011010101
1	AA111AA	89124125615

Day: Friday / Saturday — Summary: Number of Bookings: 2; Total Income: £20.00

10 This is the first problem that has occurred. A telephone number cannot be so short, “38”. The solution is to add a length check, a minimum of 7 characters in the validation, and a message box to alert the user. However, a presence check applies also to the telephone field.



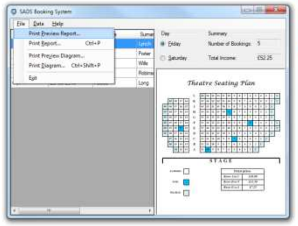
SADS Booking System

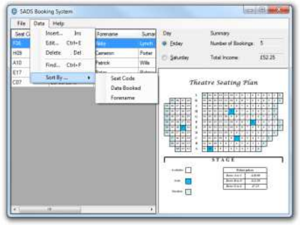
House Number	Post Code	Telephone
1A	B 344F	1011010101
1	AA111AA	38

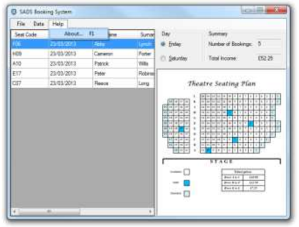
Day: Friday / Saturday — Summary: Number of Bookings: 2; Total Income: £20.00

11 The menu work effectively, as all the functionality is displayed through the menu navigation. In addition, the user can use the keyboard to access all the features, for accessibility, using:

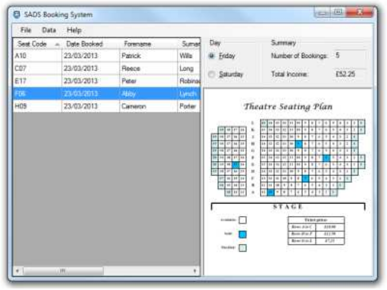
- Arrows to go to the next or previous functionality.
- Letters to access items of the selected root.
- Shortcuts access functionality without even opening the menu.







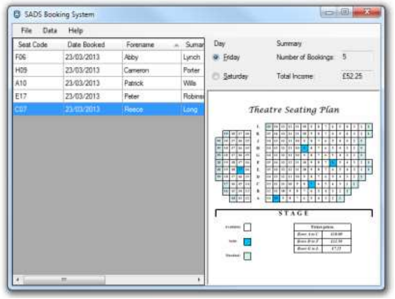
12 The sorting works fine, as A10 < C07 < E17 < F06 < H09. In addition, a bevelled arrow appears on the heading field of "Seat Code", to indicate the field has been sorted. All the items have the same date booked, so a date booked will not reorder the items. However, this sort will be in active if there are other dates booked.



Seat Code	Date Booked	Forename	Surna
A10	23/03/2013	Patrick	Wills
C07	23/03/2013	Reece	Long
E17	23/03/2013	Peter	Robinso
F06	23/03/2013	Abby	Lynch
H09	23/03/2013	Cameron	Porter

Number of Bookings: 5
Total Income: £52.25

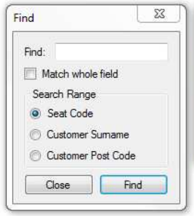
13 This sort also works well, as Abby < Cameron < Patrick < Peter < Reece.



Seat Code	Date Booked	Forename	Surna
F06	23/03/2013	Abby	Lynch
H09	23/03/2013	Cameron	Porter
A10	23/03/2013	Patrick	Wills
E17	23/03/2013	Peter	Robinso
C07	23/03/2013	Reece	Long

Number of Bookings: 5
Total Income: £52.25

14 A message box appears when there is nothing in the search textbox.


Find

Find:

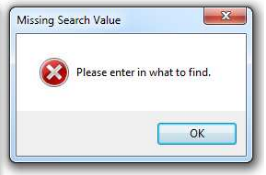
☐ Match whole field

Search Range

● Seat Code

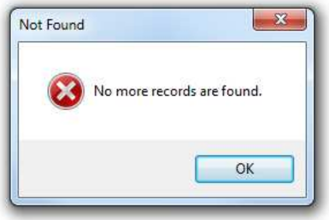
○ Customer Surname

○ Customer Post Code


Missing Search Value

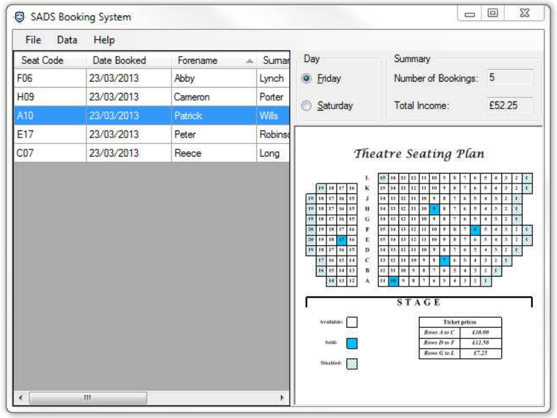
Please enter in what to find.

15 Message box appears when there is no booking that matches the search criteria.


Not Found

No more records are found.

16 The first item that matches the criteria is selected, as shown by the blue highlight, with the find dialog box.


SADS Booking System

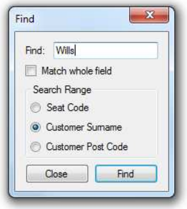
Seat Code	Date Booked	Forename	Sumar
F06	23/03/2013	Abby	Lynch
H09	23/03/2013	Cameron	Porter
A10	23/03/2013	Patrick	Wills
E17	23/03/2013	Peter	Robins
C07	23/03/2013	Reece	Long

Day: ● Friday ○ Saturday

Summary

Number of Bookings: 5

Total Income: £52.25


Find

Find: Wills

☐ Match whole field

Search Range

○ Seat Code

● Customer Surname

○ Customer Post Code

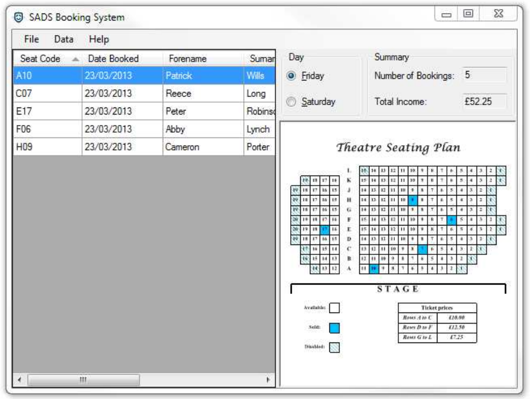
17

Seat Code	Cost
A10	£10.00
C07	£10.00
E17	£12.50
F06	£12.50
H09	£7.25
Total	**£52.25**

From this, it is clear the calculations are correct.

$$Total = \sum Seats$$

Seat	Cost
A-C	£10.00
D-F	£12.50
H-L	£7.25

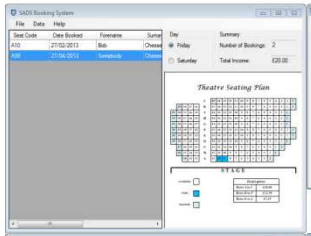
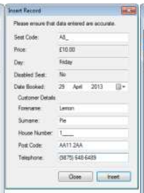

SADS Booking System

Seat Code	Date Booked	Forename	Suman
A10	23/03/2013	Patrick	Wills
C07	23/03/2013	Reece	Long
E17	23/03/2013	Peter	Robins
F06	23/03/2013	Abby	Lynch
H09	23/03/2013	Cameron	Porter

Day: [✓ Friday] Saturday

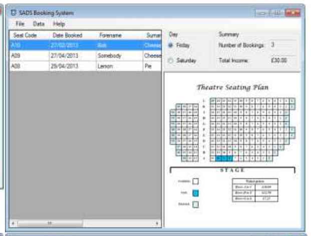
Summary
Number of Bookings: 5
Total Income: £52.25

18 From the left screenshot, there are two records, with a inset form, with the corresponding information. The right screenshot is proof that the new record is inserted corrected.

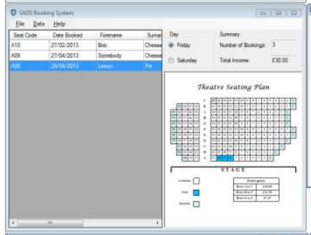
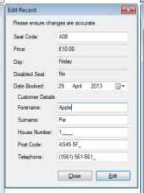


Insert Record

Please ensure that data entered are accurate.

Seat Code:	A8_
Price:	£10.00
Day:	Friday
Disabled Seat:	No
Date Booked:	29 April 2013
Customer Details	
Forename:	Lemon
Surname:	Pie
House Number:	1___
Post Code:	AA11 2AA
Telephone:	(9875) 648-6489

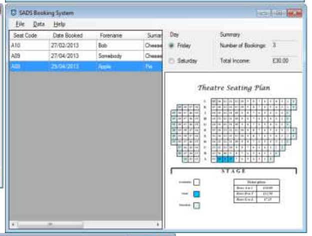


19 From the left screenshot, the third record is currently being edited. The edit dialog shows the forename has been changed. The right screenshot shows the edit being applied, this is proof that the edit feature works correctly.

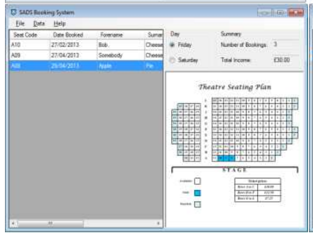
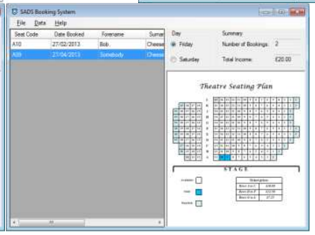


Edit Record

Please ensure changes are accurate.

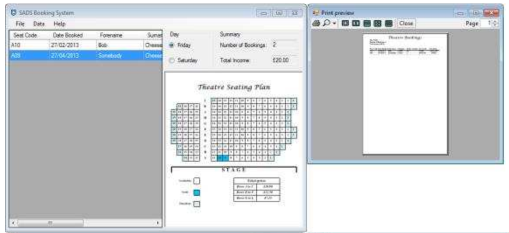
Seat Code:	A08
Price:	£10.00
Day:	Friday
Disabled Seat:	No
Date Booked:	29 April 2013
Customer Details	
Forename:	Apple
Surname:	Pie
House Number:	1___
Post Code:	AS45 5F_
Telephone:	(1561) 561-561_



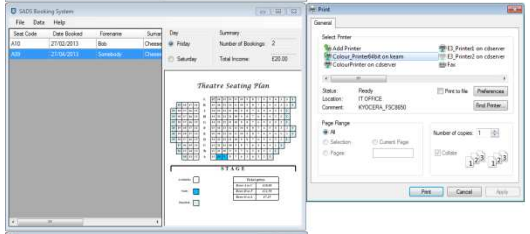
20 The left screenshot shows the Data View Grid has three bookings. The right screenshot shows two records, which clearly show that the selected booking has deleted successfully.

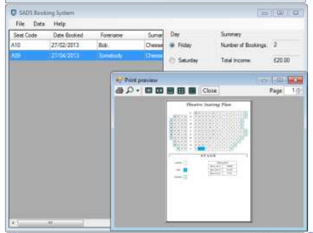
21 The print preview dialog on the right shows the same information as the Data View Grid on the left, so it the print preview works fine.



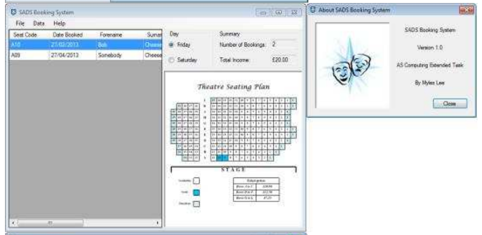
22 The print dialog appears when the button is pressed, and the printer settings are all configured correctly. Therefore, it works.



23 The print preview of the diagram works fine, as the diagram is the same as the above screenshot, but only in a print preview dialog.

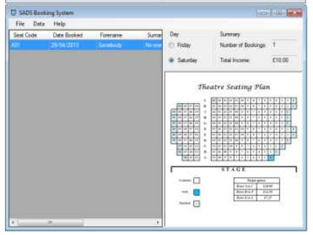


24 When the about button is clicked, the about form appears correctly.

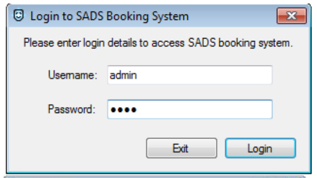


25 When the Saturday radio button is clicked, the entire form updates including the:

- Diagram
- Summary Box, including total revenue
- Data View Grid



26 The login button works and goes to the next form. The login form closes successfully, and the main form appears. The transition is smooth.



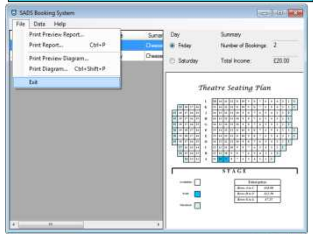
Login to SADS Booking System

Please enter login details to access SADS booking system.

Username: admin

Password: ••••

27 The exit button works, and the program, as well as the process thread that manages the program close smoothly. All the data is saved before the program closes for both tables, so there is no data loss.



SADS Booking System

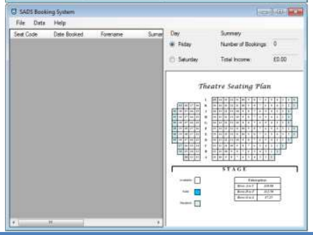
Day: [✓ Friday] Saturday

Summary

Number of Bookings: 2

Total Income: £20.00

28 When the database file is deleted to reset the table, the table recreates itself, so that further changes can be saved. This means that the user does not need to tell the program to create a new table specifically. This occurs with both the Friday and Saturday bookings tables.



SADS Booking System

Seat Code	Date Booked	Forename	Surname

Day: [✓ Friday] Saturday

Summary

Number of Bookings: 0

Total Income: £0.00

Evaluation

Usability

I aim to get at least 90% of the objectives, so that my program is suitable for client. I do this by asking several people about my program, by asking them to complete the questionnaire. This includes people outside and inside my computing class, to give a more accuracy to my results. I have given five different options, which is a good amount, without confusing people from too many options, or creating a limitation from the lack of detail from the results.

I have designed my program to work very effectively, so it should work with people who have little or have a lot of experience with computer applications.

Questionnaire Results

Name: Okantan Ayeh

Please tick the appropriate box, depending on the views of the program.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The graphical interface is appealing.					√
The program is easy to navigate.					√
The program effectively manages data.					√
The reports produced are clear.				√	
The program is robust when dealing with unexpected data.					√

Comments: Compact user interface, which makes it very nice.

Name: Ali Khaliq

Please tick the appropriate box, depending on the views of the program.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The graphical interface is appealing.					√
The program is easy to navigate.				√	
The program effectively manages data.					√
The reports produced are clear.					√
The program is robust when dealing with unexpected data.					√

Comments: Some parts of the navigation are too complex for inexperience computer users.

Name: Ake Titahmboh

Please tick the appropriate box, depending on the views of the program.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The graphical interface is appealing.				√	
The program is easy to navigate.					√
The program effectively manages data.					√
The reports produced are clear.					√
The program is robust when dealing with unexpected data.					√

Comments: The program is very effective at dealing with its task.

Name: Ammar Abdul Hadi

Please tick the appropriate box, depending on the views of the program.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The graphical interface is appealing.				√	
The program is easy to navigate.					√
The program effectively manages data.					√
The reports produced are clear.					√
The program is robust when dealing with unexpected data.					√

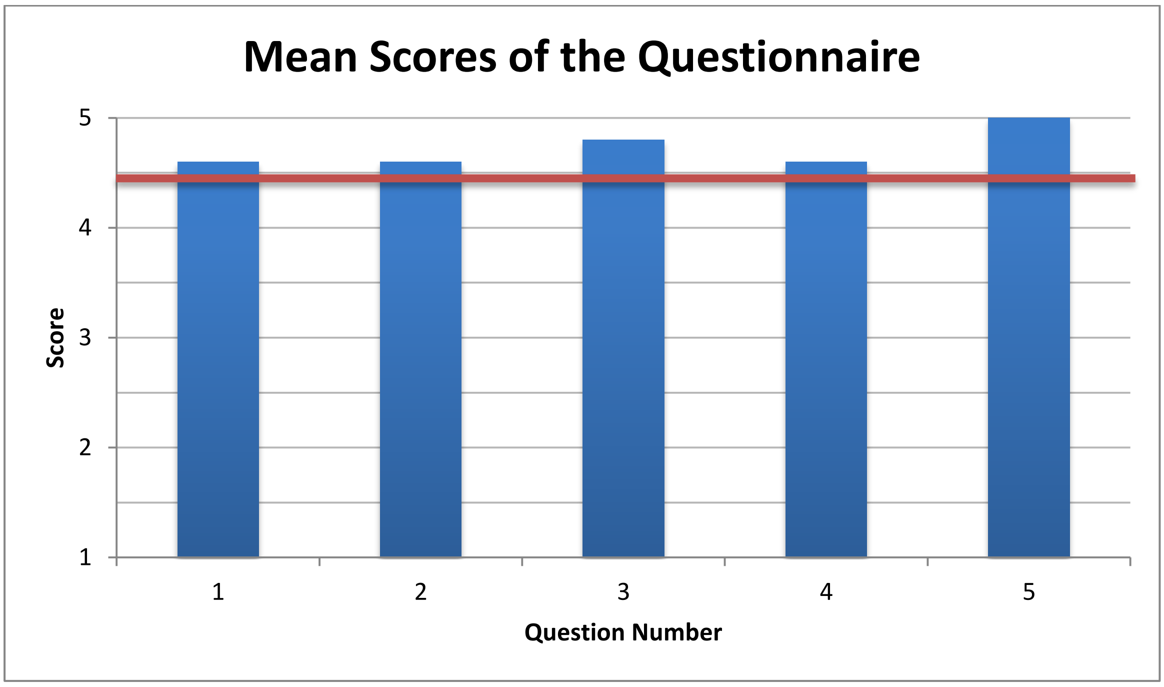
Comments :Include the user guide

Name: Sufyaan Akram

Please tick the appropriate box, depending on the views of the program.	Strongly Disagree	Disagree	Neutral	Agree	Strongly Agree
The graphical interface is appealing.					√
The program is easy to navigate.				√	
The program effectively manages data.				√	
The reports produced are clear.				√	
The program is robust when dealing with unexpected data.					√

Comments: Very intuitive graphical interface, however to improve there should be a facility to clear the table.

	Reviewer					
Question #	1	2	3	4	5	Mean Score
1	5	5	4	4	5	4.6
2	5	4	5	5	4	4.6
3	5	5	5	5	4	4.8
4	4	5	5	5	4	4.6
5	5	5	5	5	5	5



The red line shows my target score, to prove my project is effective. From the questionnaire, it is clear that all my results from the questions meet the requirements of 90% or better. Therefore, it is successful.

Suitability

My objectives as stated in the first section of Analysis and Design need to be fulfilled, since these are the primary purpose of the application. The below work effectively:

- Creation of a database for each event which will store customer booking information, which will be blank
- Records can be:
    a. added in according places because the database will sequential and input data while in the main application
    b. selected then deleted to remove records
    c. edited to make amendments and refunds
    d. retrieved to view information by either viewing the seating plan or table view
- Sorting dynamically on each heading such as names, seats in different methods such as ascending order or descending order.
- Find and replace all records
- Searching and filtering each record in the database by conditions, for example to display all available seats

- The data should store some basic information about the performance itself, such as the showing date, and the genres.
- Program will be able to cope with different formats of data so that it can widely use with other applications
- Opening and saving the database using file structures on the hard drive disk

Overall, this objective work well, and does what the end user expects, and achieves the above objectives. These reflect how successful my program is. Therefore, I would absolutely state that my program is a success.

Performance

It is important that the program works effectively, so that the software that I have designed complemented the end user's hard ware limitations. The performance of each process works very fast, often too fast for the user to detect a delay. One test in Test Runs, show the minimum memory requirements, thus showing it is suitable for most computers. However, when opening or saving files, there is a slight delay when saving to the hard drive is in use. This definitely does not last too long for the user to abort the application. This is a limitation of the .NET framework, so I am not able to fix this. Generally, the GUI gives a rapid response so that the program is continuously running.

When new bugs or an issue occurs, I can easily change my code through corrective or adaptive maintenance. Therefore was no program within my code syntax, which further improves the reliability of my project. I have also set my complier to set all warnings as errors, as well as using the highest level of warnings, to further show my program is robust.

Future Improvements

There a few features or adaptations I would like to improve my program. However, due to the lack of time to program and plan this, I was not able to do so. I would improve:

- When the user clicks on the diagram, it would select the appropriate record on the Data View Grid. This is good as the user can access records faster, making the program more effective.
- Make the interface customisable, by allowing certain items to be visible or hidden on the GUI. This would reduce the amount of clutter on the GUI, so the user does not need to be distracted with features the user will not use.
- The program should contain an interface for password recovery. Without this, the end user would not be able to access the customer booking data, which would solve the given problem. This can be done by a second password, in a section of the manual, which can change the primary password.
- Password should only be entered up to three times, or the user can use software to 'hack' the program, by using repetitive processes. When it has been entered three or more times, the program should be locked for a certain amount of time, for security purposes.
- The program should have stricter validation to further ensure accurate data. As stated before, the telephone should include a range check. Validation can also be applied on forename, by using lookup, since "environment" is not a name.