Prototype Justifications

Objective	Problems	Sub Problems	Justification
Allow the user to add a new employee to the system, entering key information including: Full name, Address, Telephone number and national insurance number. This will then be stored in a permanent file.	Produce the read back and read write routines for the files. Include the 'Add Staff' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have chosen to include adding staff to the prototype as it will be needed in other key areas of the program such as the schedule it also allows the customer to see how the staff information will be stored on the system. However, I have not chosen to include the validation of the inputs as I don't think it's necessary to the programs functioning and wasn't something the customer asked for particularly during his interview.
Permit the user to change information stored on an employee e.g., Address and telephone number.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have chosen to only include the employee telephone number to be changed as what the customer sees for both changes will be very similar so any opinions they may have on the changing of the telephone will be then same as the home address and therefore can also be implemented, including both in the prototype seems unnecessary due to the similarities. Furthermore, I will not be including the validation as I don't think it is necessary to the program functioning and wasn't something the customer asked for

			particularly during his interview.
Ability to delete a member of staff that no longer works there.	Produce the read back and read write routines for the files. Include the 'Delete Staff Member' to the menu structure. Make sure the staff member you wish to delete exists on the system before attempting to remove them.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the staff member exists by checking within the staff file.	I have chosen to include this because it allows the customer to see how staff members would be removed from the system and allows the customer to say of any changes that they would like for their system. However, I have omitted the validation as it is not necessary in ensuring the staff member is deleted and is not something that will overly impact the program functioning.
Allow the user to search for staff via reference.	Produce a routine which will search for staff members by reference. Create a routine to read back the file.	Make sure the member of staff exists. Produce a nicely formatted list of staff information.	I have chosen to include the searching of a staff member as it shows the customer how they will search for the customer and also how the information outputted is displayed this means that any issues the customer may have with what is being searched or how it looks can be tackled in the prototype rather than the end product.
Include validation on all data entered when adding a new employee.	Validate the following information: - Postcode - National Insurance number - Telephone Number	Produce routines to validate the information using parameter passing.	I have chosen to omit all validation on inputted data as it is not something that stops the program from functioning and does not take away from other areas of the program so therefore seems unnecessary to include.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	For my prototype I decided not to include backup and recovery as it does not affect the overall running of the

	Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.		program. Furthermore there will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.
Allow the user to enter a new customer to the system by storing: Full name, Address, and a basic description of the job in a permanent file.	Produce the read back and read write routines for the files. Include the 'Add Customer' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have chosen to include adding customers as it allows the customer to see how the customer information will be stored on the system and also how the stored information can be accessed and used in other aspects of the program. However, I have not chosen to include the validation of the inputs as I don't think it's necessary to the programs functioning and wasn't something the customer asked for particularly during his interview.
Allow the user to delete a customer from the system when they are no longer using the company.	Produce the read back and read write routines for the files. Include the 'Delete Customer' to the menu structure. Make sure the customer you wish to delete exists on the system before attempting to remove them.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the customer exists by checking within the customer file.	I have chosen to include this because it allows the customer to see how customers would be removed from the system and allows the customer to say of any changes that they would like for their system. However, I have omitted the validation as it is not necessary in ensuring the customer is deleted and is not something that will overly impact the program functioning.

Have the ability to change the address of a customer and telephone number stored on the system.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have chosen to only include the customer address to be changed as what the customer sees for both changes will be very similar so any opinions they may have on the changing of the address will be then same as the telephone number and therefore can also be implemented. Including both in the prototype seems unnecessary due to the similarities and also because I have included changing the staff telephone number in the prototype so they will be almost identical so it allows the customer to see both aspects. Furthermore, I will not be including the validation as I don't think it is necessary to the program functioning and wasn't something the customer asked for particularly during his interview.
Enable the user to search for customer using name or reference.	Produce a routine which will search for customers by name. Create a routine to read back the file.	Make sure the customer exists. Produce a nicely formatted list of customer information.	I have chosen to include the searching of a customer by name as it shows the customer how they will search for the customer and also how the information outputted is displayed this means that any issues the customer may have with what is being searched or how it looks can be tackled in the prototype rather than the end product. Furthermore I chose to search by name

			because in the prototype staff will be searched by reference so it means that the customer can see how both aspects of the program in regards to searching would work.
Ensure all data entered is validated.	Validate the following information: - Postcode - Telephone number	Produce routines to validate the information using parameter passing.	I have chosen to omit all validation on inputted data as it is not something that stops the program from functioning and does not take away from other areas of the program so therefore seems unnecessary to include.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	For my prototype I decided not to include backup and recovery as it does not affect the overall running of the program. Furthermore there will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.
Allow the user to add a new quote to the system by storing it in a permanent file. The information to be stored on a quote will be: Date the quote was produced, customer information, a job description, the number of days required to complete the job, a basic idea of what stock will be needed and an estimated price of the job.	Produce the read back and read write routines for the files. Include the 'Add Quote' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have chosen to include adding quotes as it allows the customer to see how the quote information will be stored on the system and also how the stored information can be accessed and used in other aspects of the program. However, I have not chosen to include the validation of the inputs as I don't think it's necessary to the programs functioning and wasn't

			something the customer asked for particularly during his interview.
Enable the user to delete a quote once the booking is finalised.	Produce the read back and read write routines for the files. Include the 'Delete Quote' to the menu structure. Make sure the quote you wish to delete exists on the system before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the quote exists by checking within the quotes file.	I have chosen to include this because it allows the customer to see how quotes would be removed from the system and allows the customer to say of any changes that they would like for their system. However, I have omitted the validation as it is not necessary in ensuring the customer is deleted and is not something that will overly impact the program functioning.
The ability to change the information on the quote including the estimated price and the number of days required.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have chosen to only include the quote price to be changed as what the customer sees for both changes will be very similar so any opinions they may have on the changing of the price will be then same as the number of days and therefore can also be implemented. Including both in the prototype seems unnecessary due to the similarities. Furthermore, I will not be including the validation as I don't think it is necessary to the program functioning and wasn't something the customer asked for particularly during his interview.
Enable the user to search for a specific quote using	Produce a routine which will search for a quote by quote	Make sure the quote exists.	I have chosen to include the searching of a quote by reference as

quote reference or customer reference.	reference or customer reference. Create a routine to readback the file.	Produce a nicely formatted list of details of the quote.	it shows the customer how they will search for the quote and also how the information outputted is displayed this means that any issues the customer may have with what is being searched or how it looks can be tackled in the prototype rather than the end product. Furthermore I chose to only search by 1 field as it will be very similar processes for both so multiple searched would be unnecessary in a prototype.
Produce a calculation for the quote taking into account current stock prices, labour costs, number of days worked and mileage.	Produce a multistage calculation using different mathematical operators and parameter passing to return the value of the calculation.	Produce a format to show the stages of the calculation broken down including VAT.	I have chosen to include the calculation of the quote as it is one of the most important aspects because the whole system is based around quotes and bookings. Furthermore it also allows the customer to see how the stock prices will be automatically linked to the calculation so the customer can confirm if this will be useful and is done in a way they like and can give appropriate feedback.
Validate all quote information upon entry.	Validate the following information: - Quote reference	Produce routines to validate the information using parameter passing.	I have chosen to omit all validation on inputted data as it is not something that stops the program from functioning and does not take away from other areas of the program so therefore seems unnecessary to include.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data.	Produce validation routine to ensure the level of access before	For my prototype I decided not to include backup and recovery as

	Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	allowing data to be backed up.	it does not affect the overall running of the program. Furthermore there will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.
Allow the user to add additional stock to a permanent file including information such as: colour of paint, type of paint, volume of can, price and quantity.	Produce the read back and read write routines for the files. Include the 'Add Stock' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have chosen to include adding stock as it allows the customer to see how the stock information will be stored on the system and also how the stored information can be accessed and used in other aspects of the program. However, I have not chosen to include the validation of the inputs as I don't think it's necessary to the programs functioning and wasn't something the customer asked for particularly during his interview.
Allow the user to delete stock from the system if it is no longer required or is no longer sold.	Produce the read back and read write routines for the files. Include the 'Delete Item of Stock' to the menu structure. Make sure the item of stock you wish to delete exists on the system before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the item of stock exists by checking within the stock file.	I have chosen to include this because it allows the customer to see how items of stock would be removed from the system and allows the customer to say of any changes that they would like for their system. However, I have omitted the validation as it is not necessary in ensuring the customer is deleted and is not something that will overly impact the program functioning.

The ability to change the price and quantity of the stock.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	I have chosen to only include the stock quantity to be changed as what the customer sees for both changes will be very similar so any opinions they may have on the changing of the price will be then same as the number of days and therefore can also be implemented. Including both in the prototype seems unnecessary due to the similarities. Furthermore, I will not be including the validation as I don't think it is necessary to the program functioning and wasn't something the customer asked for particularly during his interview.
Enable the user to search for stock using stock ID to display information.	Produce a routine which will search for stock by stock ID. Produce a routine to read back the file.	Make sure the item of stock exists. Produce a nicely formatted list of stock information.	I have chosen to include the searching of an item of stock as it shows the customer how they will search for the stock and also how the information outputted is displayed this means that any issues the customer may have with what is being searched or how it looks can be tackled in the prototype rather than the end product.
Enable the user to sort stock quantities from low to high.	Produce a routine which will sort stock quantities. Have the option to sort stock on the user interface.	Make sure there is stock to sort. Create a program to read back and rewrite to the file.	I have chosen to include the sort in the prototype because it is the only one so there will be no other chance for the customer to critique this in any way until the final product therefore I thought it

			was important to include.
Produce a message when stock quantities drop to a specific level.	Carry out a loop which will produce a message when stock reaches a certain level.	Make sure the loop will stop running after it has checked through all the stock and only produce the message when there is stock below the quantity specified.	I have chosen to include this because it means that the customer can decide if how the message is displayed is useful and if there are any changes they would wish for.
Ensure information entered is validated.	Validate the following information: - Price - Quantity	Produce routines to validate the information using parameter passing.	I have chosen to omit all validation on inputted data as it is not something that stops the program from functioning and does not take away from other areas of the program so therefore seems unnecessary to include.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	For my prototype I decided not to include backup and recovery as it does not affect the overall running of the program. Furthermore there will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.
Allow the user to add a new booking to the schedule including: Customer information and the start date of the job and stores this in a permanent text file.	Produce the read back and read write routines for the files. Include the 'Add to schedule' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have chosen to include adding to the schedule as it allows the customer to see how the bookings will be displayed on the system. However, I have not chosen to include the validation of the inputs as I don't think it's necessary to the programs functioning and wasn't something the customer asked for

			particularly during his interview.
Allow only those with the highest access to change information such as date of the job.	Produce the read back and read write routines for the files. Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered. Use validation to check the access level before allowing the change to happen.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists. Produce a message if the level of access is invalid.	I have chosen to include the date of the booking to be changed so the customer can see how this affects the schedule when viewing it and allows the customer to say of any changes they would like on how the data on the schedule is changed. Furthermore, I will not be including the validation as I don't think it is necessary to the program functioning and wasn't something the customer asked for particularly during his interview. There will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.
Have the ability to delete information such as a cancelled booking.	Produce the read back and read write routines for the files. Include the 'Remove Cancellation' to the menu structure. Make sure the booking you wish to delete exists on the system before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the quote exists by checking within the quote file.	I have chosen to include this because it allows the customer to see how cancellations would be removed from the system and allows the customer to say of any changes that they would like for their system. However, I have omitted the validation as it is not necessary in ensuring the customer is deleted and is not something that will overly impact the program functioning.
Include validation on the new information which is entered including the date.	Validate the following information: - Date	Produce routines to validate the	I have chosen to omit all validation on inputted data as it is

	- Quote reference - Staff number	information using parameter passing.	not something that stops the program from functioning and does not take away from other areas of the program so therefore seems unnecessary to include.
Ensure there is hierarchical access with levels of access of the administrator and the members of staff below.	Produce a program using parameter passing to check whether a staff member has the ability to access and edit the schedule.	Create messaged to be outputted to the screen to tell a user if they do not have access.	There will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	For my prototype I decided not to include backup and recovery as it does not affect the overall running of the program. Furthermore there will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.
Allow the user to add a new invoice to the system storing this in a permanent text file including the following information: Customer information, Date of the job, Description of the job and a new price.	Produce the read back and read write routines for the files. Include the 'Add Invoice' to the menu structure. Use a program to input the required information. Validation on all inputs.	Use validation routines and parameter passing to check the data inputted. Make sure that the menu structure is clear and easy to read.	I have chosen to include adding invoices as it allows the customer to see how the invoices will be stored on the system. However, I have not chosen to include the validation of the inputs as I don't think it's necessary to the programs functioning and wasn't something the customer asked for particularly during his interview.
Allow the user to be able to change information including date of the job	Produce the read back and read write routines for the files.	Use validation routines and parameter passing to check the data inputted.	I have chosen to only include the invoice payment status because this will be

and whether the job has been paid for.	Include the 'Change Information' to the menu structure. Use a program to input the required information. Re-validate the information entered.	Make sure that the menu structure is clear and easy to read. Search to find the information you wish to change exists.	used in other aspects of the program. Also, the changing of the dates will be very similar so any opinions they may have on the changing of the payment status will then be same as the dates and therefore can also be implemented. Including both in the prototype seems unnecessary due to the similarities. Furthermore, I will not be including the validation as I don't think it is necessary to the program functioning and wasn't something the customer asked for particularly during his interview.
The ability to delete an invoice from the system once it is no longer needed and the job has been paid.	Produce the read back and read write routines for the files. Include the 'Delete Invoice' to the menu structure. Make sure the invoice you wish to delete exists on the system / that the file is not empty before attempting to remove it.	Make sure that the menu structure and delete section is clear and easy to understand. Use validation routine to ensure the invoice exists by checking within the invoices file.	I have chosen to include this because it allows the customer to see how invoices would be removed from the system and allows the customer to say of any changes that they would like for their system. However, I have omitted the validation as it is not necessary in ensuring the customer is deleted and is not something that will overly impact the program functioning.
Produce a multistage calculation for the invoice taking into account: number of days worked, mileage costs, current stock costs and labour costs.	Produce a multistage calculation using different mathematical operators and parameter passing to return the value of the calculation.	Produce a format to show the stages of the calculation broken down including VAT.	I have chosen to include the calculation as it is one of the most important aspects since this will be what is given to the customer at the end. It is included in the prototype because the

			customer will be most familiar with how the calculation takes place manually so will be able to give feedback to any changes on the calculation they would like before it reaches the final design.
Enable the user to be able to search for an invoice by invoice or customer reference.	Produce a routine which will search for invoices by either references. Use a readback routine.	Make sure the invoice exists. Produce a nicely formatted list displaying details of an invoice.	I have chosen to include the searching of an invoice by customer reference as it shows the customer how they will search for the invoice and also how the information outputted is displayed this means that any issues the customer may have with what is being searched or how it looks can be tackled in the prototype rather than the end product. Furthermore, I chose to search by customer reference because it demonstrates to the customer how different aspects of the company can link together and also because I have included a search by staff reference which would be very similar to a search by invoice reference.
Enable the user to be able to search for paid or unpaid invoices.	Produce a routine which will search for the invoices by whether they have been paid or not. Use a readback routine.	Produce a list of invoices that are currently unpaid.	I have chosen to include this search as it is unique in regards to the other searches included in the prototype therefore it allows the customer to see a different aspect of being able to view certain criteria and they will be able to give feedback on anything they dislike in this as it I

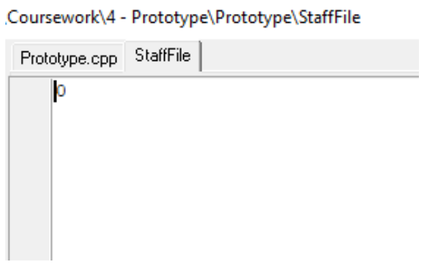
			the only time it is used in the program so if it wasn't included and the customer disliked the feature it wouldn't be found out until the end of the project.
Ensure there is validation in place for all information entered excluding the description of the job.	Validate the following information: - Invoice reference - Staff number - Price - Customer reference	Produce routines to validate the information using parameter passing.	I have chosen to omit all validation on inputted data as it is not something that stops the program from functioning and does not take away from other areas of the program so therefore seems unnecessary to include.
Ensure all information saved is backed up.	Create a routine to allow the backing up of data. Produce read back and rewrite routines for the file. Use security to allow only certain staff to back up data.	Produce validation routine to ensure the level of access before allowing data to be backed up.	For my prototype I decided not to include backup and recovery as it does not affect the overall running of the program. Furthermore there will be no security included in the prototype as it does not affect the backbone of the system that the customer will be assessing once given the prototype.

User Interface

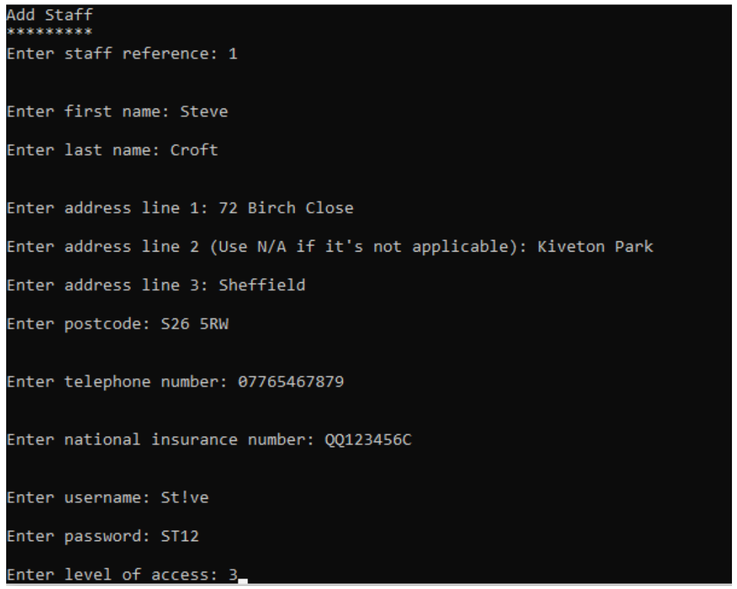

Main Menu
****
1. Staff
2. Customers
3. Quotes
4. Invoices
5. Stock
6. Schedule
7. Quit
Enter choice: 1
Staff Menu
****
1. Add Staff
2. Change staff information
3. Delete staff
4. View staff
5. Backup
6. Recovery
7. Return to Main Menu
Enter choice: 2
Change Staff Information
****
1. Change Home Address
2. Change Telephone Number
3. Change Username
4. Change Password
5. Change Level of Access
6. Return to Staff Menu
Enter choice:
Customer Menu
****
1. Add Customer
2. Change customer information
3. Delete customer
4. View customer
5. Return to Main Menu
Enter choice: 2
Change Customer Information
****
1. Change home address
2. Change telephone number
3. Return to Customer Menu
Enter choice:
View Customer Information
****
1. Search customer by name
2. Search customer by customer reference
3. Return to Customer Menu
Enter choice:
Quotes Menu
****
1. Add Quote
2. Change quote information
3. Delete quote
4. View quote
5. Return to Main Menu
Enter choice:
Change Quote Information
****
1. Change quote price
2. Change number of days required
3. Return to Quote Menu
Enter choice:
View Quotes
****
1. Search quote by quote reference
2. Search quote by customer name
3. Return to Quote Menu
Enter choice:
Invoice Menu
****
1. Add Invoice
2. Change invoice information
3. Delete invoice
4. View invoice
5. Return to Main Menu
Enter choice:
Change Invoice Information
****
1. Change dates worked
2. Change payment status
3. Return to Invoice Menu
Enter choice:
View Invoices
****
1. Search invoice by invoice refernece
2. Search invoice by customer reference
3. Search for unpaid invoices
4. Return to Invoice Menu
Enter choice:
Stock Menu
****
1. Add Stock
2. Change stock information
3. Delete stock
4. View stock
5. Return to Main Menu
Enter choice:
Change Stock Information
****
1. Change price of stock
2. Change stock quantities
3. Return to Stock Menu
Enter choice:
View Stock Information
****
1. Search stock by stock ID
2. Sort stock quantities low to high
3. Return to Stock Menu
Enter choice:
Schedule Menu
****
1. Add Booking
2. Change booking information
3. Delete booking
4. View schedule
5. Clear all schedule
6. Return to Main Menu
Enter choice:
View Schedule
****
1. View Staff 1
2. View Staff 2
3. Return to Schedule Menu
Enter choice:

Staff File

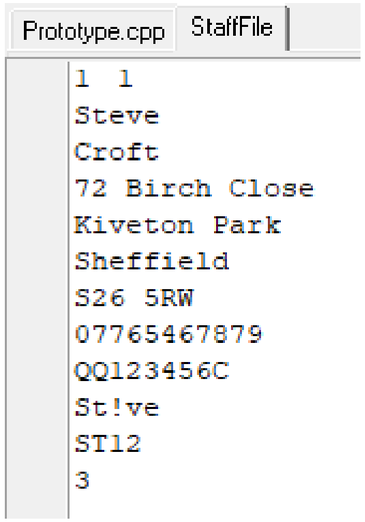
Adding a member of staff to the staff file:



Shows the empty staff file



Shows the employee's information being added



Shows the inputted information in the staff file

Viewing a member of staff from the staff file:

```
View Staff Member by Staff Reference
************************************
Enter staff reference: 1

Name: Steve Croft

Address: 72 Birch Close
         Kiveton Park
         Sheffield
         S26 5RW

Telephone number: 07765467879

National Insurance Number: QQ123456C

Level of access: 3
```

Shows the user viewing a member of staff by entering the staff reference

Prototype.cpp | StaffFile

```
1 1
Steve
Croft
72 Birch Close
Kiveton Park
Sheffield
S26 5RW
07765467879
QQ123456C
St!ve
ST12
3
```

Shows the outputted information in the staff file

Changing the telephone number of a staff member:

Prototype.cpp | StaffFile

```
3 1
Steve
Croft
72 Birch Close
Kiveton Park
Sheffield
S26 5RW
07765467879
QQ123456C
St!ve
ST12
3
2
Lois
Bridge
42 Oak Close
North Anston
Sheffield
S25 3SR
07654758650
AA12564V
L2
charl!e
2
3
Steph
Smith
12 William Road
N/A
Sheffield
S1 4RT
07654098767
WW12546R
SS
```

Shows the original staff file before any changes

```
Enter the staff reference of the staff member's telephone number you wish to change: 1
Enter new mobile number: 07234017306
```

Shows the user changing the mobile number of the employee with staff reference 1 – “Steve Croft”

Prototype.cpp | StaffFile

```
3 1
Steve
Croft
72 Birch Close
Kiveton Park
Sheffield
S26 5RW
07234017306
QQ123456C
St!ve
ST12
3
2
Lois
Bridge
42 Oak Close
North Anston
Sheffield
S25 3SR
07654758650
AA12564V
L2
charl!e
2
3
Steph
Smith
12 William Road
N/A
Sheffield
S1 4RT
07654098767
WW12546R
SS
```

Shows the updated mobile number in the staff file

Deleting a member of staff from the staff file:

Prototype.cpp | StaffFile

```
3  1
Steve
Croft
72 Birch Close
Kiveton Park
Sheffield
S26 5RW
07234017306
QQ123456C
St!ve
ST12
3
2
Lois
Bridge
42 Oak Close
North Anston
Sheffield
S25 3SR
07654758650
AA12564V
L2
charl!e
2
3
Steph
Smith
12 William Road
N/A
Sheffield
S1 4RT
07654098767
WW12546R
SS
```

Shows all the staff currently stored in the file

```
Enter the staff reference of the staff member you wish to delete: 1
```

Shows the user deleting a member of staff by entering the staff reference

Prototype.cpp | StaffFile

```
2  2
Lois
Bridge
42 Oak Close
North Anston
Sheffield
S25 3SR
07654758650
AA12564V
L2
charl!e
2
3
Steph
Smith
12 William Road
N/A
Sheffield
S1 4RT
07654098767
WW12546R
SS
S1
1
```

Shows the updated version of the staff file once the user deleted the staff member with the staff reference 1 – “Steve Croft”

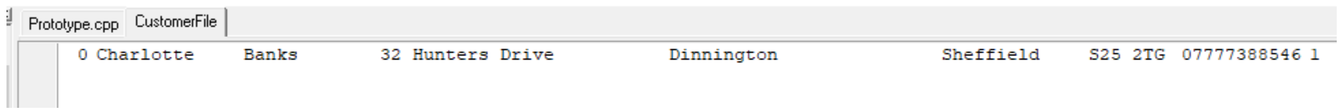
Customer File

Adding a customer to the customer file:

Shows the empty customer file

```
Add Customer************
Enter new customer reference: 1

Enter first name: Charlotte

Enter last name: Banks


Enter address line 1: 32 Hunters Drive

Enter address line 2: Dinnington

Enter address line 3: Sheffield

Enter postcode: S25 2TG


Enter mobile number: 07777388546
```

Shows the customers information being added



Shows the inputted information in the customer file

Viewing a customer from the customer file:

```
View Customer by Customer Name
******************************
Enter customers last name: O'Neil

Full name: Claire O'Neil

Address line 1: 10 Nether Oak Close
Address line 2: Sothall
Address line 3: Sheffield
Postcode: S20 2PT

Telephone number: 07765297370
```

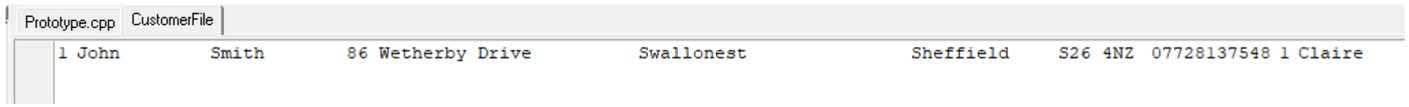
Shows the user viewing a customer by entering the customers last name

Prototype.cpp | CustomerFile

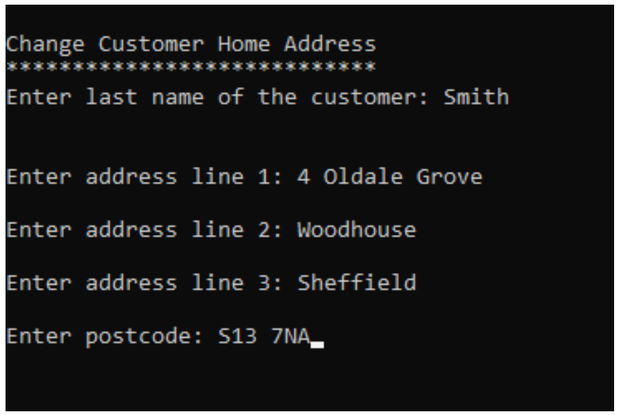
```
1 Claire       O'Neil       10 Nether Oak Close        Sothall                Sheffield     S20 2PT  07765297370 1
```

Shows the outputted information in the customer file

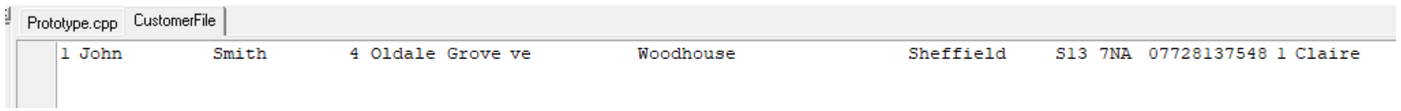
Changing the home address of a customer:



Shows the original customer file before any changes

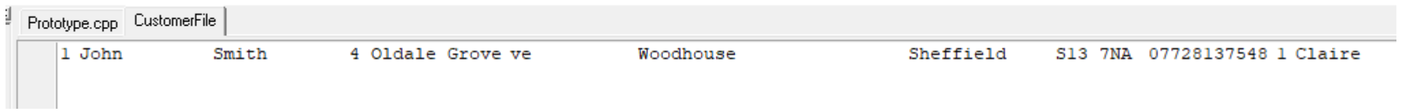


Shows the user changing the customers address

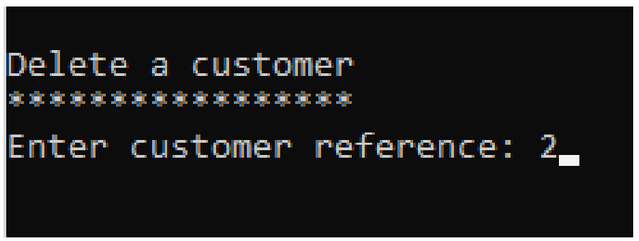


Shows the updated address in the customer file

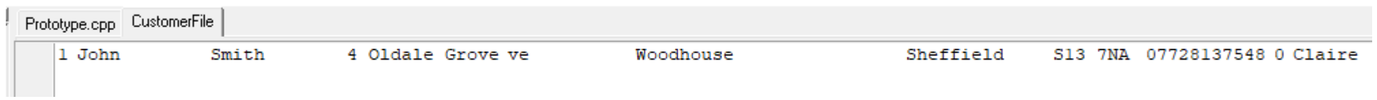
Deleting a customer from the customer file:



Shows all the customers currently stored in the folder the highlighted ‘1’ represents the flag is 1 as there is a customer stored at that position.



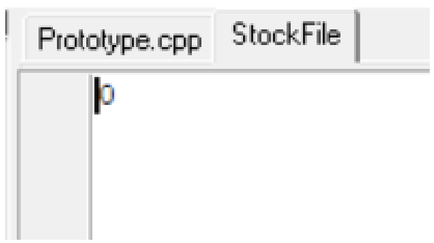
Shows the user deleting a customer by entering the customer reference



Shows the deleted customer and the highlighted flag has now changed to ‘0’ to show the flag is empty, the customer has been deleted, and the space is free to use again.

Stock File

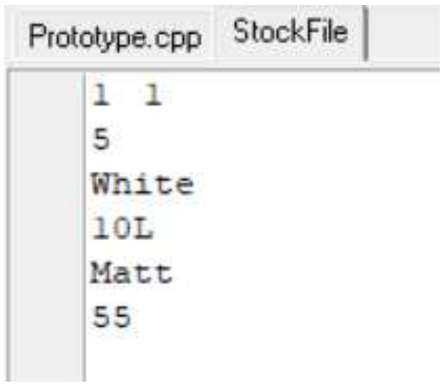
Adding an item of stock to the stock file:



Shows the empty stock file

```
Add Stock
*********
Enter stock reference: 1

Enter colour: White

Enter type of paint: Matt

Enter stock price: 55

Enter volume: 10L

Enter quantity: 5
```

Shows the stock information being added



Shows the inputted information in the stock file

Viewing an item of stock from the stock file:

```
View Stock by Stock Reference
*****************************
Enter stock reference: 1

Colour: White
Stock Price 55
Quantity: 5
```

Shows the user viewing an item of stock by entering the stock reference

Prototype.cpp StockFile

```
3  1
5
White
10L
Matt
55
2
3
Black
25L
Matt
30
3
12
Green
15L
Gloss
39
```

Shows the outputted information in the stock file

Sorting stock quantities from the stock file:

Prototype.cpp | StockFile

```
3  1
10
White
10L
Matt
55
2
3
Black
25L
Matt
30
3
12
Green
15L
Gloss
39
```

Stock file showing unsorted stock information in the file

```
Sorted Stock Quantities:

Stock Reference: 2
Quantity: 3

Stock Reference: 1
Quantity: 10

Stock Reference: 3
Quantity: 12
```

Provides a list of sorted stock quantities alongside stock references

Changing the quantity of an item of stock:

Prototype.cpp | StockFile

```
3  1
5
White
10L
Matt
55
2
3
Black
25L
Matt
30
3
12
Green
15L
Gloss
39
```

Shows the original stock file before any changes

```
Change stock quantity
*********************
Enter the stock reference of the stock items quantity you want to change: 1

Enter new quantity: 1
```

Shows the user changing the quote of the item of stock with stock reference 1

Prototype.cpp | StockFile

```
3  1
1
White
10L
Matt
55
2
3
Black
25L
Matt
30
3
12
Green
15L
Gloss
39
```

Shows the updated quantity in the stock file

Deleting an item of stock from the stock file:

Prototype.cpp | QuotesFile2

```
3  1
1
Paint, Kitchen, Blue
7
1050
35

282
3000
2
2
Paint, living room, cream
4
600
20

255
1575
3
1
paint, bathroom, black
3
450
12
```

Shows all the stock currently stored in the file

```
Delete quote
************
Enter the quote reference of the quote you wish to delete: 1
```

Shows the user deleting an item of stock by entering the stock reference

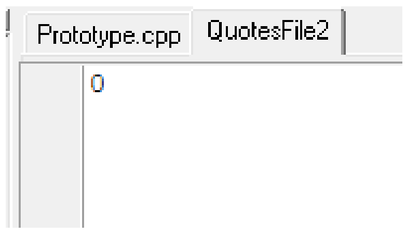
Prototype.cpp | QuotesFile2

```
2  2
2
Paint, living room, cream
4
600
20

255
1575
3
1
paint, bathroom, black
3
450
12

133
1071
```

Shows the updated version of the stock file once the user deleted the stock with the stock reference 1

Quotes File

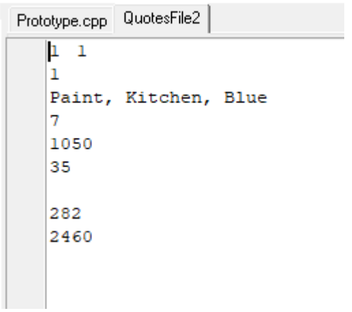
Adding a quote to the quotes file:



Shows the empty quotes file

```
Add Quote
*********
Enter quote reference: 1

Enter customer reference: 1


Enter job description: Paint, Kitchen, Blue


Enter number of days on the job: 7

Enter travel costs: 35

How many items of stock are required: 3
1
Enter stock reference: 1

Stock Price: 45
What quantity is required: 2
1
Enter stock reference: 2

Stock Price: 60
What quantity is required: 1
1
Enter stock reference: 3

Stock Price: 44
What quantity is required: 3

Price Breakdown
***************
Materials: 282
Labour: 1050
Mileage: 35
VAT: 1093
Total: 2460
```

Shows the quote information being added



Shows the inputted information in the quotes file

The multistage calculation for the quote:

```
Add Quote
*********
Enter quote reference: 1

Enter customer reference: 1


Enter job description: Paint, Kitchen, Blue


Enter number of days on the job: 7

Enter travel costs: 35

How many items of stock are required: 3
1
Enter stock reference: 1

Stock Price: 45
What quantity is required: 2
1
Enter stock reference: 2

Stock Price: 60
What quantity is required: 1
1
Enter stock reference: 3

Stock Price: 44
What quantity is required: 3

Price Breakdown
***************
Materials: 282
Labour: 1050
Mileage: 35
VAT: 1093
Total: 2460
```

Shows the information being inputted by the user for the quote calculation.

Labour costs: (150*7). The number of days on the job multiplied by the standard day rate.

Materials costs found by finding the stock cost for the item multiplying it by the quantity needed. Uses a running total for all items of stock required to work out the total cost of materials.

The total cost is calculated by adding together all the separate calculations including VAT.

Prototype.cpp | QuotesFile2

```
1  1
1
Paint, Kitchen, Blue
7
1050
35

282
2460
```

Shows the results of the calculation from the users inputs.

Viewing a quote from the quote file:

```
View Quote by Quote Reference
*****************************
Enter quote reference: 1

Customer reference: 1
 Customer Name Charlotte Banks
 Customer Telephone Number 07777388546

Job Description: Paint, Kitchen, Blue

Number of days worked: 7

Price Breakdown
***************
Materials: 282
Labour: 1050
Mileage: 35
Total Cost: 2460
```

Shows the user viewing a quote by entering the quote reference

Prototype.cpp | QuotesFile2

```
3  1
1
Paint, Kitchen, Blue
7
1050
35

282
2460
```

Shows the outputted information in the quote file

Changing the price of a quote:

Prototype.cpp | QuotesFile2

```
3  1
1
Paint, Kitchen, Blue
7
1050
35

282
2460
2
2
Paint, living room, cream
4
600
20

255
1575
3
1
paint, bathroom, black
3
450
12
```

Shows the original quote file before any changes

```
Change price of quote
*********************
Enter the quote reference of the quote price you want to change: 1

Enter new price: 3000
```

Shows the user changing the price of the quote with quote reference 1

Prototype.cpp | QuotesFile2

```
3  1
1
Paint, Kitchen, Blue
7
1050
35

282
3000
2
2
Paint, living room, cream
4
600
20

255
1575
3
1
paint, bathroom, black
3
450
12
```

Shows the updated price in the quote file

Deleting a quote from the quote file:

Prototype.cpp | QuotesFile2

```
3  1
1
Paint, Kitchen, Blue
7
1050
35

282
3000
2
2
Paint, living room, cream
4
600
20

255
1575
3
1
paint, bathroom, black
3
450
12
```

Shows all the quotes currently stored in the file

```
Delete quote
************
Enter the quote reference of the quote you wish to delete: 1
```

Shows the user deleting a quote by entering the quote reference

Prototype.cpp | QuotesFile2

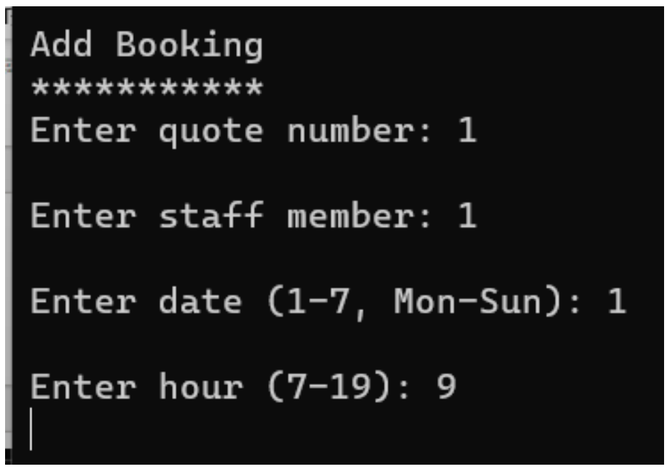
```
2  2
2
Paint, living room, cream
4
600
20

255
1575
3
1
paint, bathroom, black
3
450
12

133
1071
```

Shows the updated version of the quote file once the user deleted the quote with the quote reference 1

Schedule File

Adding a booking to the schedule file:

Shows the empty schedule file. Asterisk represents an empty space on the schedule where a booking can be added.
Schedule file is represented as one straight line so this shows the start of the file to prove it is empty



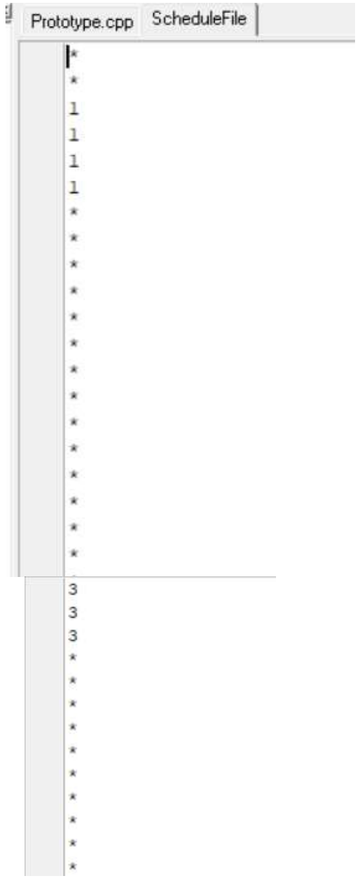
Shows the booking information being added

Shows the inputted information in the schedule file represented by the quote reference.

Viewing a booking from the schedule file:

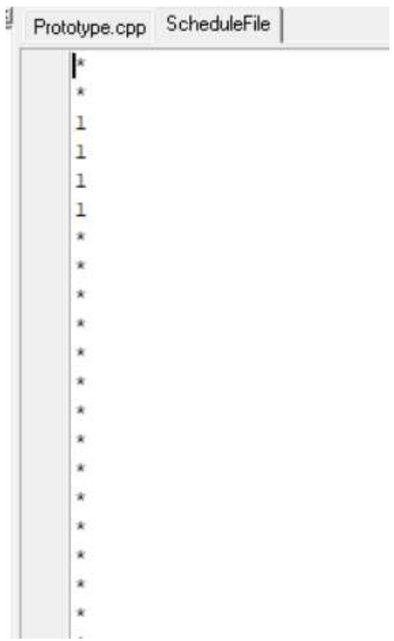
```
View Booking
 Staff Reference: 1
       07      08      09      10      11      12      13      14      15      16      17      18      19
Mon    *       *       1       1       1       1       *       *       *       *       *       *       *
Tue    *       *       *       *       *       *       *       *       *       *       *       *       *
Wed    3       3       3       *       *       *       *       *       *       *       *       *       *
Thu    *       *       *       *       *       *       *       *       *       *       *       *       *
Fri    *       *       *       *       *       *       *       *       *       *       *       *       *
Sat    *       *       *       *       *       *       *       *       *       *       *       *       *
Sun    *       *       *       *       *       *       *       *       *       *       *       *       *
```

Shows the user viewing a booking by looking at the correct staff member



Shows the outputted information in the schedule file

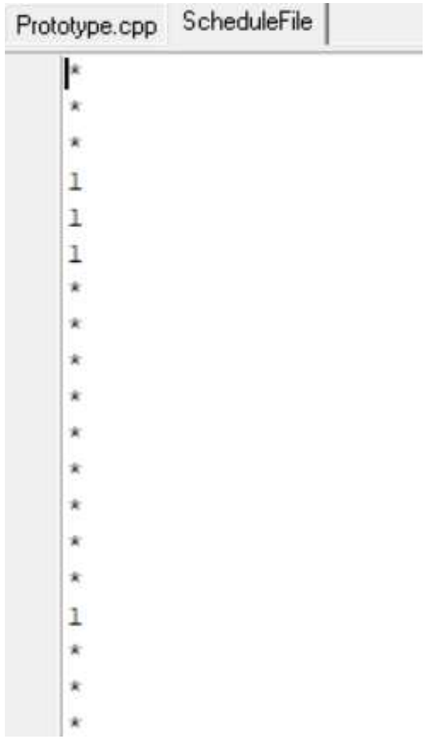
Changing the date of a booking:



Shows the original schedule file before any changes

```
Change date of booking
**********************
 Enter staff member: 1

 Enter old date of booking(1-7): 1

 Enter old time of booking(7-19): 9

 Enter new date of booking(1-7): 2

Enter quote reference: 1
```

Shows the user changing the date of the booking.



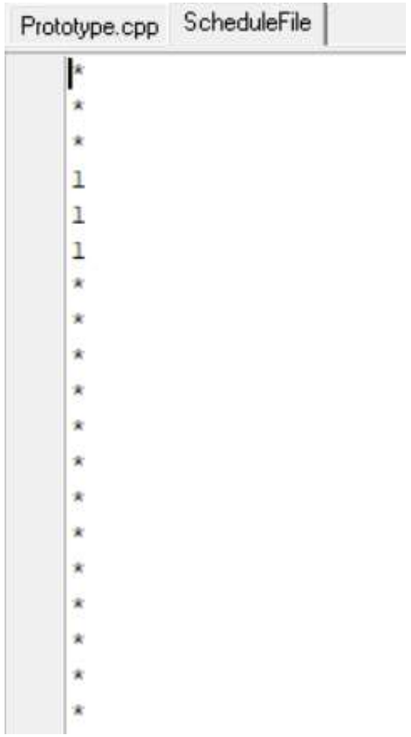
Shows the updated booking in the schedule file

Deleting a booking from the schedule file:

Shows all the bookings currently stored in the file

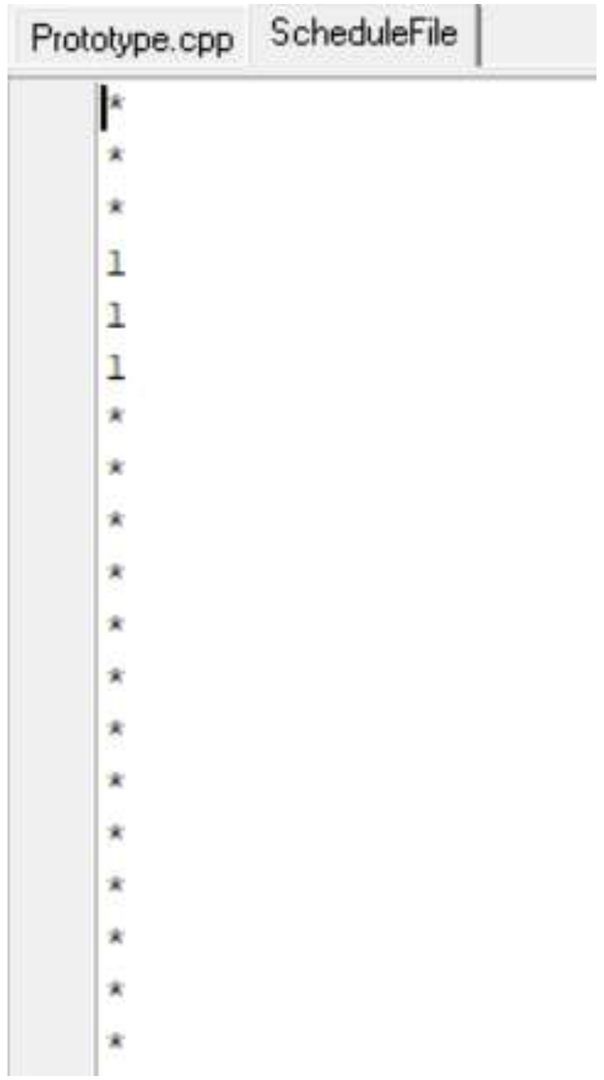
```
Delete Booking
**************
Enter the staff reference of the booking that has been cancelled: 1

Enter the day that the booking has been cancelled (1-7): 2

Enter the hour of the booking that has been cancelled: 9
```

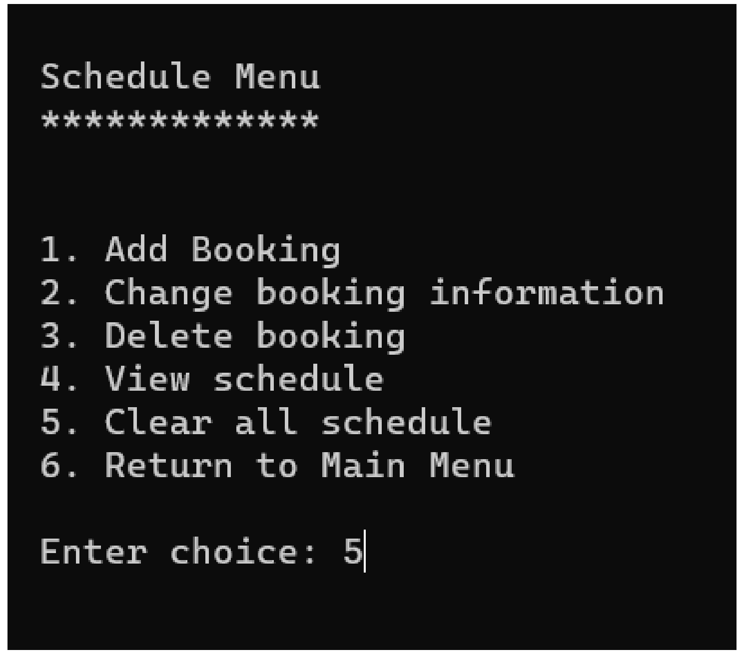
Shows the user deleting a booking



Shows the updated version of the schedule file once the user deleted the booking – the quote reference has been replaced with an asterisk

Clearing all bookings from the schedule file:





User selects the option to clear the schedule

Show the updated schedule file now that all bookings have been removed

Shows the original schedule file

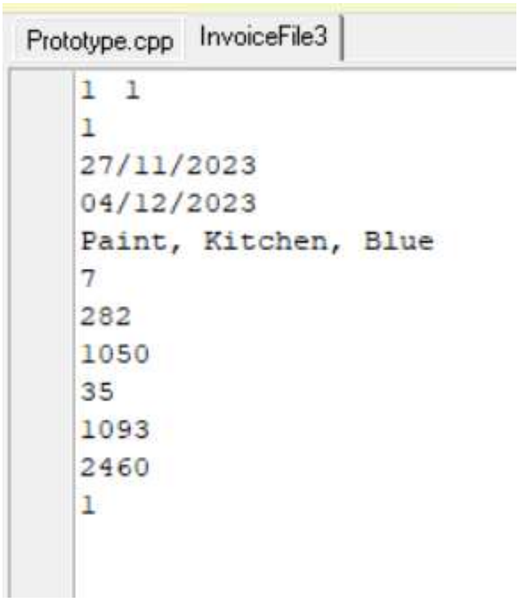
Invoice File

Adding an invoice to the invoice file:

Shows the empty invoice file

```
Add Invoice
***********
Enter invoice reference: 1

Enter customer reference: 1


Enter start date: 27/11/2023

Enter end date: 04/12/2023


Enter job description: Paint, Kitchen, Blue


Enter number of days worked: 7

Enter cost of materials: 282

Enter travel costs: 35


Is the job paid for:
(O=Yes, 1=No)1

Price Breakdown
***************
Materials: 282
Labour: 1050
Mileage: 35
VAT: 1093
Total: 2460
```

Shows the invoice information being added



Shows the inputted information in the invoice file

The multistage calculation for the invoice:

```
Add Invoice
***********
Enter invoice reference: 1

Enter customer reference: 1


Enter start date: 27/11/2023

Enter end date: 04/12/2023


Enter job description: Paint, Kitchen, Blue


Enter number of days worked: 7

Enter cost of materials: 282

Enter travel costs: 35


Is the job paid for:
(0=Yes, 1=No)1

Price Breakdown
***************
Materials: 282
Labour: 1050
Mileage: 35
VAT: 1093
Total: 2460
```

Shows the information being inputted by the user for the quote calculation.

Labour costs: (150*7). The number of days on the job multiplied by the standard day rate.

VAT calculated by adding together the total and multiplying it by 0.8 to find the 20% vat rate to add to the final price

The total cost is calculated by adding together all the separate calculations including VAT.

Shows the results of the calculation from the users inputs.

Viewing an invoice from the invoice file:

```
View by customer reference
*************************
Enter customer reference: 1

Customer reference: 1
Customer Name: Charlotte Banks
Customer Telephone Number: 07777388546

Job Description: Paint, Kitchen, Blue

Dates worked: 27/11/2023-04/12/2023

Price Breakdown
***************
Materials: 282
Labour: 1050
Mileage: 35
VAT: 109
Total: 2460
```

Shows the user viewing an invoice by entering the customer reference

Prototype.cpp | InvoiceFile3

```
3 1
1
27/11/2023
04/12/2023
Paint, Kitchen, Blue
7
282
1050
35
109
2460
1
2
2
12/03/2024
15/03/2024
Paint, Bathroom, Black
4
350
600
12
769
1731
0
3
3
```

Shows the outputted information in the invoice file

Viewing all unpaid invoices from the invoice file:

```
View Unpaid Invoices
********************
Customer reference: 1
Customer reference: 3
```

Shows the customer references of the invoices that have not yet been paid.

Prototype.cpp | InvoiceFile3

```
3  1
1
27/11/2023
04/12/2023
Paint, Kitchen, Blue
7
282
1050
35
109
2460
1
2
2
12/03/2024
15/03/2024
Paint, Bathroom, Black
4
350
600
12
769
1731
0
3
3
01/01/2024
05/01/2024
Paint, Stairs, White
5
150
750
10
728
1638
1
```

Shows the invoice file with relevant payment statuses

Changing the payment status of an invoice:

Prototype.cpp | InvoiceFile3

```
3  1
1
27/11/2023
04/12/2023
Paint, Kitchen, Blue
7
282
1050
35
109
2460
1
2
2
12/03/2024
15/03/2024
Paint, Bathroom, Black
4
350
600
12
769
1731
0
3
3
```

Shows the original invoice file before any changes

```
Change Payment Status
*********************
Enter the invoice reference of the invoice you want to change: 1

Enter new payment status: 0
```

Shows the user changing the payment status using the invoice reference

Prototype.cpp | InvoiceFile3

```
3  1
1
27/11/2023
04/12/2023
Paint, Kitchen, Blue
7
282
1050
35
109
2460
0
2
2
12/03/2024
15/03/2024
Paint, Bathroom, Black
4
350
600
12
769
1731
0
3
3
```

Shows the updated payment status in the invoice file

Deleting an invoice from the invoice file:

Prototype.cpp | InvoiceFile3

```
3  1
1
27/11/2023
04/12/2023
Paint, Kitchen, Blue
7
282
1050
35
109
2460
0
2
2
12/03/2024
15/03/2024
Paint, Bathroom, Black
4
350
600
12
769
1731
0
3
3
```

Shows all the invoices currently stored in the file

```
Delete Invoice
**************
Enter the invoice reference of the invoice you wish to delete: 1
```

Shows the user deleting an invoice by entering the invoice reference

Prototype.cpp | InvoiceFile3

```
2  2
2
12/03/2024
15/03/2024
Paint, Bathroom, Black
4
350
600
12
109
1731
0
3
3
01/01/2024
05/01/2024
Paint, Stairs, White
5
150
750
10
769
1638
1
```

Shows the updated version of the invoice file once the user deleted the invoice with the invoice reference 1

Section	What did I do?	What was good about it?	Justification of good point	Suggested improvement	Refined design requirement
STAFF					
Add Staff Member	Created a menu structure that followed a logical order allowing the user to access a staff menu where a member of staff can be added. Following on, I created the staff file (a serial file) which would allow the information entered to be stored. Read Back and Rewrite Files were produced so that the file could be accessed but also so new inputted data can be written to the file. I programmed a suitable UI with outputs to the screen allowing the user to input details line by line.	The logical order of the menu structure made it easy to find within the system. The outputs to the screen when adding an employee.	I believe that the order of the menu was a good point because it means that the end user would be able to quickly and easily find what they were looking for from the first use without any complications. I think that this was a good aspect of the program because it means that when the user is entering a member of staff to the system it ensures all the correct data is entered and in the correct format for when a member of staff is searched making it quicker and easier to display the employee information		
Change the telephone number of a staff member.	Created a menu structure that followed a logical order allowing the user to access a staff menu which displayed an area where staff information can be changed including the telephone number. I used the Read back and Re Write functions to firstly find the			When the change function is outputted to the screen there is only one line telling the end user to enter the number however there is no further instruction on whether to enter a	Change the design of the outputs to demonstrate the format in which way the employee number should be entered.

	staff member in the file then write the new phone number to the file. To find the correct staff member I searched through all the employees in the file until I found the staff reference that matched the one entered by the user.	Ensuring the user enters the staff reference to find and change the telephone number.	I believe that this a good point because the staff reference is unique to each employee so when it is entered the end user has the peace of mind that it is the correct employee making the section of programming more reliable. Since the staff reference is also only 1 character it means the change can happen quickly which is important when running a business.	mobile number or a telephone number and which one is the correct format.	
Delete staff member	Created a menu structure that followed a logical order allowing the user to access a staff menu where a member of staff can be deleted. An output to the screen prompts the user to enter the reference of the employee who is to be deleted. The Read back of the function means that the staff file can be read and the reference entered searched for through the file until there is a match. Re Write function allows all the private data stored about the individual to be deleted from the	Similar to above, the use of the unique staff reference.	I believe that this is a good point because it allows the end user to know they are deleting the correct employee since only they have that reference helping make the system reliable and efficient.		

	file so it can no longer be accessed.				
Search for staff via reference.	Created a menu structure that followed a logical order allowing the user to access a staff menu where a member of staff can be viewed. Read back function allows the staff file to be read through for the search. Outputs all relevant information regarding a staff member to the screen in an ordered way splitting the information up.	The use of a serial file when searching for an employee. The outputs to the screen when the staff member is found.	I believe this is a good point since the company has only a couple of employees the search will be quick to do making it very useful for the end user when they may need to quickly access a piece of information regarding an employee. I think that this is a good aspect of the program as it makes the results of the search easy to read and understand as it is split up with blank space so that it is clearly laid out.	When the user enters a reference to be deleted if the staff reference cannot be found and the employee does not exist within the system a message could be outputted to the screen to alert the user of why the search is not working rather than a blank screen.	Change the outputs of the design to show how a simple message could be outputted to the screen.
CUSTOMER					
Add Customer	Created a menu structure that followed a logical order allowing the user to access a customer menu where a customer can be added. I also created the customer file (a random access file) which would allow the information entered to be stored. I programmed it so that there were suitable outputs to make	The menu structure which allows the user to find the section of the program they require.	I think that this is a good aspect of adding a customer since it is easy to use, this means that when an employee is on the phone and quickly needs to find the correct section to enter a new customer's details its very efficient and doesn't slow down the process for either the employee or the customer.		

	sure all the information needed was entered so it could be stored in the file.	The outputs to the screen when adding customer information.	As the employee is likely to be on the phone to the customer regarding the booking they have forgotten to write something down when trying to be quick. However, the prompts on screen resolve this issue and ensure the employee retrieves all the information they will require about a customer in the future whilst also quickly adding the customer to the system.		
Delete Customer	Created a menu structure that followed a logical order allowing the user to access a customer menu where a customer can be deleted. An output to the screen prompts the user to enter the reference of the employee who is to be deleted. As it is a random file the flags value would be changed to zero.	Similar to the staff, making the user enter the customer reference to find which to delete.	I think that this is good because each customer is assigned a unique reference this means that when a customer is deleted from the system the user knows exactly who it is and the whole process is clear and defined making it stress free and easy to use.		
Change customer home address via last name	Created a menu structure that followed a logical order allowing the user to access a customer menu which displayed an area where customer information can be changed including their home address. As it is a random file once the customer reference is entered it				

	can jump straight to the customer within the file ready to update it with the new information. Outputs lines of the address one at a time.	The outputs to the screen to aid the changing of the address.	By outputting prompts that allows the user to input the new address it means that it is easy to use and ensures the address is entered correctly I believe that this makes it a good point. Furthermore, it means that the data is stored in an organised and structured way which makes viewing much easier.		
Search for customer via last name	Created a menu structure that followed a logical order allowing the user to access a customer menu which displayed an area where customer information can be viewed via last name. Used a random access file to search through customers comparing the name entered with the last names stored in the file until there was a match – this then meant the information on that customer could be outputted. Outputs useful information to the screen about a customer in a split up way so that it is presented nicely.	Use of last name to search for a customer Use of random file for searches.	I believe that searching by the last name is a good point because if an employee needs to find a customer they are likely to know the name of that customer rather than a number (the reference) this means that the usability of the search facility may increase compared to a search via a reference. I believe using a random access file is a good point for the customer file because a random file allows fast searching since it has the ability to jump straight to a customer whereas a serial would have to load the whole file. This means that even when the file size is large with many customers it can still efficiently locate the customer.	If a customer's last name is forgotten or is misspelled, then the customer's information would not be able to view. To stop this user should also have the option to view a customer by the reference in case the problem above occurs.	

QUOTES					
Add Quote	Created a menu structure that followed a logical order allowing the user to access a quotes menu from the main menu where a quote can be added. Following on, I created the quote file (a serial file). Read Back and Rewrite Files were produced so that the file could be accessed but also so new inputted data can be written to the file. I programmed a suitable UI with outputs to the screen allowing the user to input details line by line and leaving spaces to block off entries so that general information was together then the calculation information. Outputted the results of the calculation in a broken down format.	Programming the outputs in a way that separated the different aspects of a quote. The calculation being outputted to the screen.	I think that this is a good point because it means that the user can organise themselves ready to enter information since if they are on the phone to the customer it allows information to be entered in a structured way and allows the user to discuss each section of the quote with them. I think that the calculation being displayed at the end is a good point because when the user tells the customer the price they can further explain how this prices is derived as a clear break down shows how it is all calculated meaning the employee is knowledgeable and demonstrates to the customer the reliably of the price since it shows it per section.		
Delete Quote	Created a menu structure that followed a logical order allowing the user to access a quote menu from the main menu where a quote can be deleted. An output to the screen prompts the user to enter the reference	Similar to above, the use of the unique quote reference.	I believe that this is a good point because it allows the end user to know they are deleting the correct quote since only they have that reference helping make the system more reliable and efficient.		

	of the quote which is to be deleted. The Read back function means that the quote file can be read and the reference entered searched for through the file until there is a match. Re Write function allows all the data stored about the quote to be deleted from the file so it can no longer be accessed.				
Change Quote Price	Created a menu structure that followed a logical order allowing the user to access a quotes menu from the main menu which displayed an area where quote information can be changed including the price of the quote. I used the Read back and Re Write functions to firstly find the quote in the file then write the new price to the file. To find the correct quote reference I searched through all the quotes in the file until I found the quote reference that matched the one entered by the user.	Changing the price by the quote reference.	I think that this is a good point because each reference is unique this means that the system is more reliable since the changing of the price may only effect that one customer depending on materials used so it ensures no other customers are affected.	When the user inputs the updated price there is no guidance on the format which is needed for it to work correctly.	Change the design to show the user that the number entered must not include a decimal point and therefore be an integer.
Search for quote via	Created a menu structure that followed a logical order allowing				

quote reference	the user to access a quote menu from the main menu where a quote can be viewed. Read back function allows the quote file to be read through for the search. Outputs all relevant information regarding a quote to the screen in an ordered way splitting the information up with the price breakdown. Links the customer information regarding the specific quote from the customer file and outputs the most useful information.	The linking of the customer information.	I believe that this is a good point because it makes the system much more useful for the employee because when they are finding a quote they may need to contact the customer regarding something in particular this means that there is less time wasted having to search for the customer since the name and contact information is there already. Furthermore, if the quote was created a long time ago the user could have forgotten which customer this quote belonged to so by linking the customer information it helps to remind the user.		
Produce a multistage calculation	Uses the inputs from the user when adding the quote to calculate the cost. Converts	Automatic link of stock prices.	I believe that this is a good point because it increases the usability of the system since it does not		

	characters entered by the user to integer values. Locates the stock prices from the stock file using the stock references entered by the user and a links file to calculate total price of materials taking into account the quantities using a running total.		rely on the employee to know all the prices of the items of stock making the calculation much easier for the user and also quicker since they do not need to find out all the prices.		
STOCK					
Add Stock	Created a menu structure that followed a logical order allowing the user to access a stock menu where an item of stock can be added. Following on, I created the stock file (a serial file) which would allow the information entered to be stored. Read Back and Rewrite Files were produced so that the file could be accessed but also so new inputted data can be written to the file. I programmed a suitable UI with outputs to the screen allowing the user to input details line by line.	Outputs to the screen which prompts the user to enter different information regarding the item of stock.	I think that this is a good point because it makes the system easier to use because it helps the user to understand what information they need to enter at each step ensuring all the correct data is saved for the stock for uses in calculations or to simply view an item of stock.		

Delete Stock	Created a menu structure that followed a logical order allowing the user to access a stock menu where an item of stock can be deleted. An output to the screen prompts the user to enter the reference for the item of stock which is to be deleted. The Read back of the function means that the stock file can be read and the reference entered searched for through the file until there is a match. Re Write function allows all the data stored about the item of stock to be deleted from the file so it can no longer be accessed.	Prompting the user to enter the unique reference for the stock to be deleted.	I believe that this is a good point because it allows the end user to know they are deleting the correct item of stock since it's the only stock with that reference helping make the system reliable and efficient.		
Change Stock Quantities	Created a menu structure that followed a logical order allowing the user to access a stock menu which displayed an area where stock information can be changed including the quantity of an item of stock. I used the Read back and Re Write functions to firstly find the item of stock in the file then write the quantity to the file. To find the correct item of stock I searched through all the stock in the file until I found the stock	Similar to above, making the user enter the unique reference.			

	reference that matched the one entered by the user.				
Search for stock via stock reference	Created a menu structure that followed a logical order allowing the user to access a stock menu where an item of stock can be viewed. Read back function allows the stock file to be read through for the search. Outputs all relevant information regarding an item of stock to the screen in an ordered way splitting the information up.	Outputs to the screen	I believe that this is a good point because it increases the readability of the screen since the different pieces of information are split up with spaces aiding the user rather than all information grouped together at the top with little to no indication of what it means.		
Sort stock quantities low to high	Created a menu structure which gives the option to sort stock under the viewing menu. Uses the read back function to load the stock file and retrieve stock reference and quantities. I then passed them as parameters to a bubble sort which sorted the stock quantities from low to high and outputted them along with their stock references in this sorted order.	Outputting of stock references along with quantities.	I believe that this is a good point because it makes the system much more useful since if it just displayed the quantities it would		

			be useless for the staff as they would not know which item of stock it was regarding whereas if the stock reference is also outputted it means that the staff can then go further and find out what the item of stock is to determine where they need to buy more or not. However, it is also useful that the quantities are outputted since if it was just the references the user would still not know what the smallest quantity was so it could be a high number but it's just the smallest in the file so in that respect outputting the quantities is essential and could save the company money.		
Message when stock quantities drop to 0.	Uses the read back function to load the stock file and search each item of stocks quantity and if it is equal to 0 output the stock reference and the quantity of 0 to the screen when the stock menu is loaded.	Appearance with stock menu.	I think that this is a good point because it increases the efficiency for the staff since it means they are quickly alerted if certain stock levels are too low rather than having to either view stock individually or sort them.	Since currently this only shows stock with quantities of 0 I will change this to show stock with quantities less than 2 as it is more useful for the company to make sure they organised rather than when it is too late and they have already run out.	Change code to make it output stock that has a quantity of 2 or less.

SCHEDULE					
Add Booking	Created a menu structure that followed a logical order allowing the user to access a schedule menu where a booking can be added. Following on, I created the schedule file (a sequential file) which would allow the information entered to be stored. Read Back and Rewrite Files were produced so that the file could be accessed but also so new inputted data can be written to the file. Allows the user to enter the information with prompts on the correct way to enter the data.	Prompts to the user	I think that this is a good point because it makes the add booking function a lot more easy to use for the user because it tells them exactly what to put to enter the correct data ensuring that the booking is added to the file smoothly.		
Delete Booking	Created a menu structure that followed a logical order allowing the user to access a schedule menu where a booking can be deleted. An output to the screen prompts the user to enter the staff				

	member, date and time of the booking to be deleted. The schedule is then read back and the quote reference on the schedule at that specific point is replaced with an asterisk to show that this booking has been cancelled and is now available to book.	Replacement of the quote reference	I think this is a good point because it means the user doesn't have to edit the schedule in any way to show that there was a cancellation as the quote reference is automatically removed from the schedule making the process a lot quicker and easier for the staff as there is less to do.		
Change date of booking	Created a menu structure that followed a logical order allowing the user to access a schedule menu which displayed an area where schedule information can be changed including the date of the booking. I used the Read back and Re Write functions to firstly find the correct position in the array from the information entered by the user then delete the old booking date and enter the new date to the schedule.	The deleting of the old booking from the schedule.	I think that this is a good point because it increases efficiency since the user only has to access one aspect of the program rather than having to delete the old booking then add the new change.		
View Schedules for staff	Created a menu structure that followed a logical order allowing the user to access a schedule menu where an employee's schedule for the week can be viewed	Use of quote reference	I think that this is a good point because it means that when the user looks at the schedule it helps them to see who is actually booked for that time since they can go further and find the quote	The schedule only shows one week at a time however I think it would be more beneficial to be able to view 1 month at a	Amend the design so that it shows one month at a time rather than per week.

	Outputs the times and days of the week with the quote reference on the schedule to show where this is a booking.		to see more information rather than just displaying a different character.	since it's likely that one job could take the whole week this also means they would be able to book further in advance	
Clearing the schedule	Created a menu structure that followed a logical order allowing the user to access a schedule menu where the schedules can be cleared. Uses the re write functions to find every position in the array and replace any quote references with an asterisk and save these changes.	Replacement of quote references	I believe that this is a good point because it means that the staff can efficiently remove all bookings from the schedule after the week so they can restart the next week without having to individually remove each booking.		
INVOICES					
Add Invoice	Created a menu structure that followed a logical order allowing the user to access an invoice menu from the main menu where an invoice can be added. Following on, I created the invoice file (a serial file). Read Back and Rewrite Files were produced so that the file could be accessed but also so	Outputs to the screen which prompts the user to enter different information regarding an invoice.	I think that this is a good point because it makes the system easier to use because it helps the user to understand what information they need to enter at each step ensuring all the correct data is saved for the invoice for uses in calculations or for when an invoice is being viewed.		

	new inputted data can be written to the file. I programmed a suitable UI with outputs to the screen allowing the user to input details line by line and leaving spaces to block off entries so that general information was together then the calculation information. Outputted the results of the calculation in a broken down format.	The breakdown of the price being outputted.	I think that the calculation being displayed at the end is a good point because when the user tells the customer the price they can further explain how this prices is derived as a clear break down shows how it is all calculated meaning the employee is knowledgeable and demonstrates to the customer the reliably of the price since it shows it per section.		
Delete invoice	Created a menu structure that followed a logical order allowing the user to access an invoice menu from the main menu where an invoice can be deleted. An output to the screen prompts the user to enter the reference of the invoice which is to be deleted. The Read back function means that the invoice file can be read and the reference entered	Prompting the user to enter the unique reference for the invoice to be deleted.	I believe that this is a good point because it allows the end user to know they are deleting the correct invoice since it's the only invoice with that reference helping make the system reliable and efficient.		

	searched for through the file until there is a match. Re Write function allows all the data stored about the invoice to be deleted from the file so it can no longer be accessed.				
Change payment status of an invoice	Created a menu structure that followed a logical order allowing the user to access an invoice menu from the main menu which displayed an area where invoice information can be changed including the payment status of the invoice. I used the Read back and Re Write functions to firstly find the invoice in the file then write the updated payment status. To find the correct invoice reference I searched through all the invoices in the file until I found the invoice reference that matched the one entered by the user.	Similar to above, making the user enter the unique reference.			
Search for invoice via customer reference	Created a menu structure that followed a logical order allowing the user to access an invoice menu from the main menu where an invoice can be viewed. Read back function allows the invoice file to be read through for the search.	Use of customer reference	I think that this is a good point because the customer reference will have been used throughout the booking including on the quote this means that the customer reference is more likely to be remembered by the staff		

	Outputs all relevant information regarding the invoice to the screen in an ordered way splitting the information up with the price breakdown. Links the customer information regarding the specific invoice from the customer file and outputs the most useful information.	Linking of the customer information	helping make the system easier to use. I believe that this is a good point because it makes the system much more efficient for the employee because when they are finding an invoice they may need to contact the customer regarding something in particular this means that there is less time wasted having to search for the customer since the name and contact information is there already.		
Search for unpaid invoices	Created a menu structure that followed a logical order allowing the user to access an invoice menu from the main menu where an invoice can be viewed. Read back function allows the invoice file to be read through for the search and finds the invoices with an unpaid payment status then outputs the customer references of the unpaid invoices to the screen.	Outputting the customer reference.	I think that this is a good point because if the booking is still not paid for rather than showing the invoice reference the customer reference is showed because it increases the efficiency of the system since the user is able to then go and find the customer contact information if needed.	To improve this, instead of only outputting the customer reference it could also output the customer name and telephone number so that the user has to do no more to even find the customer	Amend the design so that the customer name and telephone number is also outputted under the customer reference.
Produce a multistage calculation	Uses the inputs from the user when adding the invoice to calculate the cost.	Using user inputs	I believe that this is a good point because it increases the efficiency of the system since all the user has to do is enter the values		

	Converts characters entered by the user to integer values.		rather than before when they had to manually work it out.		